# On Learning Parametric Dependencies from Monitoring Data

Johannes Grohmann, Simon Eismann, Samuel Kounev

Symposium on Software Performance (SSP) 2019

05.11.2019

https://se.informatik.uni-wuerzburg.de/

# Software Performance Models



- Performance models are a common approach to predict software performance

**Server system**

# Software Performance Models



- Performance models are a common approach to predict software performance

**Server system**

maximize

**Efficiency**

# Software Performance Models

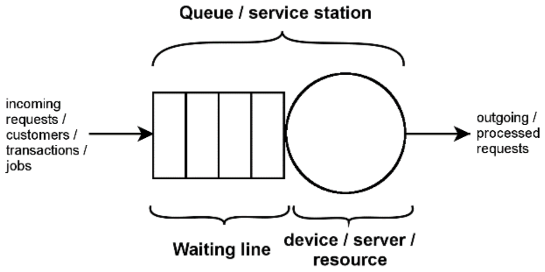


- Performance models are a common approach to predict software performance

**Server system**

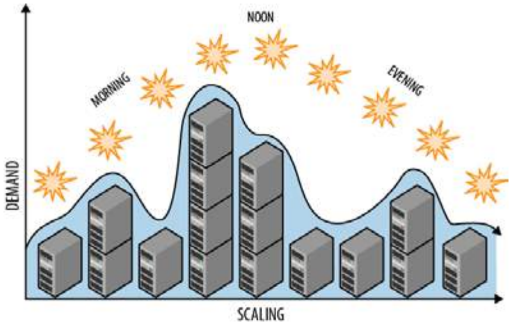
maximize

**Efficiency**




**Performance model**

e.g.




**Auto-scaler**

# Software Performance Models

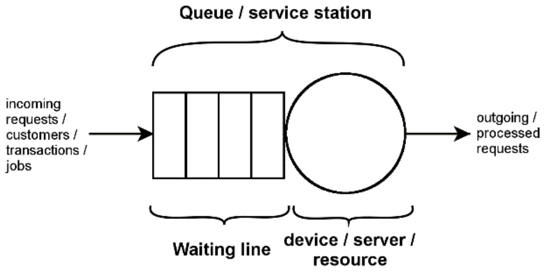


- Performance models are a common approach to predict software performance
- However, correctly modeling a software system is difficult

**Server system**

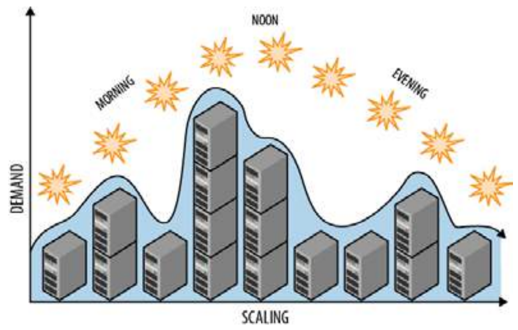
maximize

**Efficiency**




**Performance model**

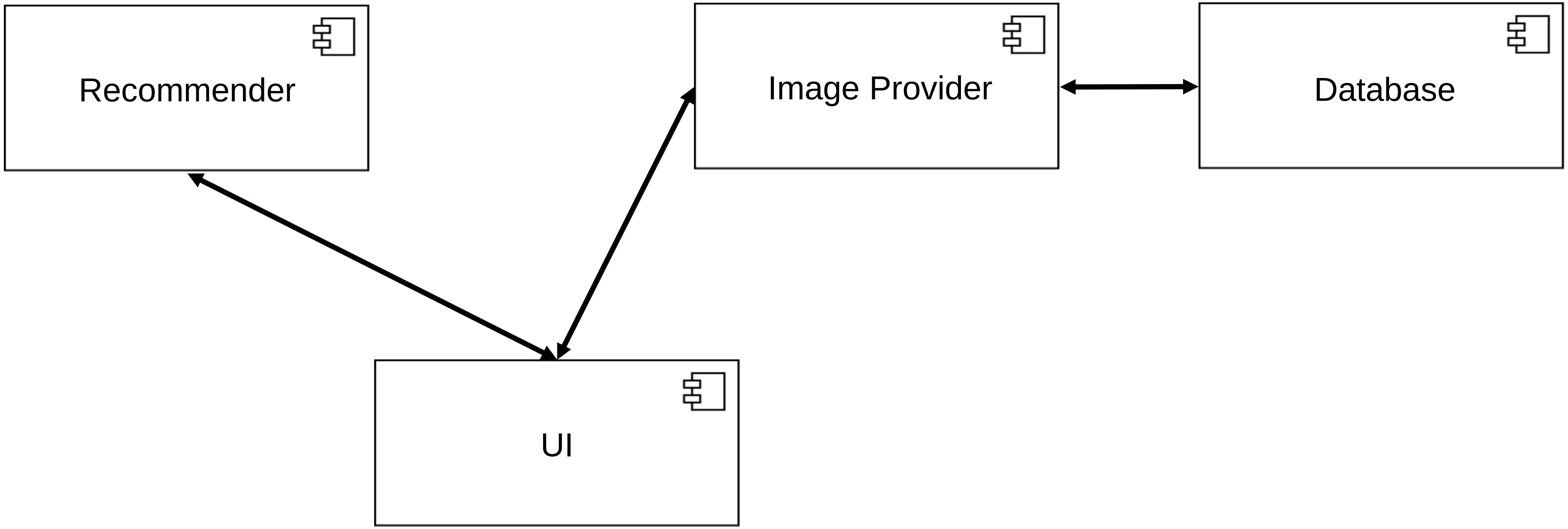
e.g.




**Auto-scaler**

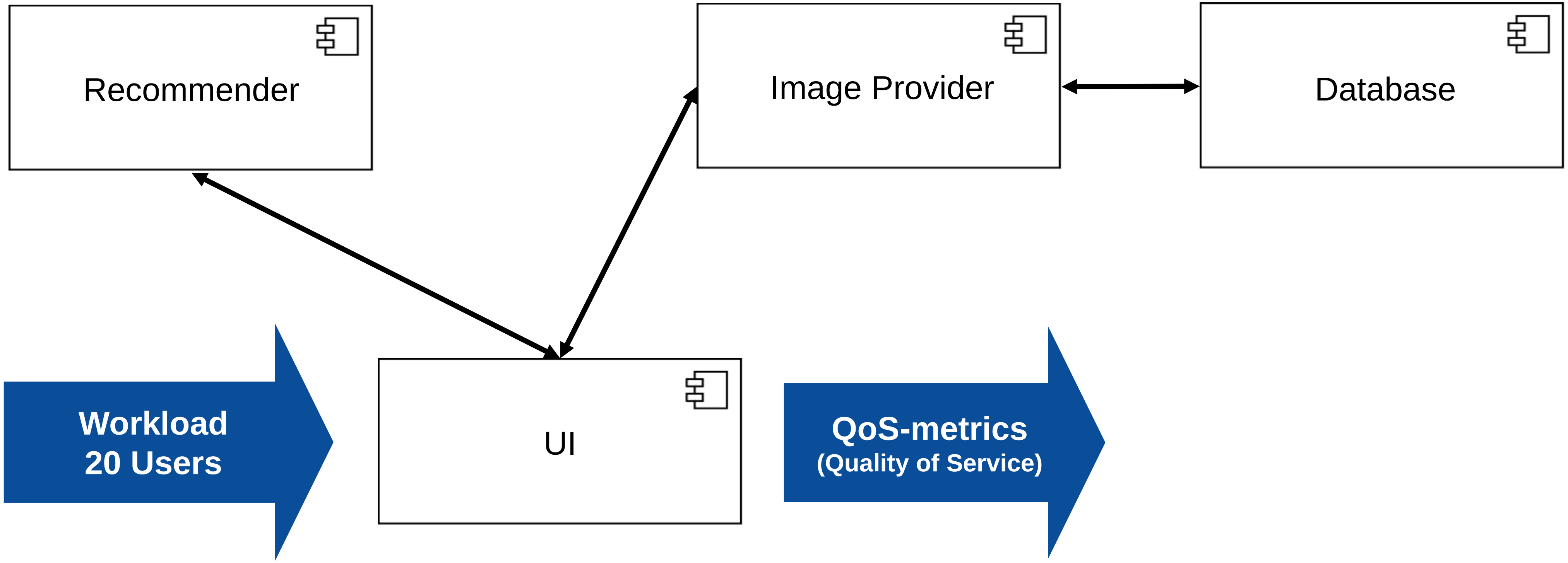
# Parametric Dependencies





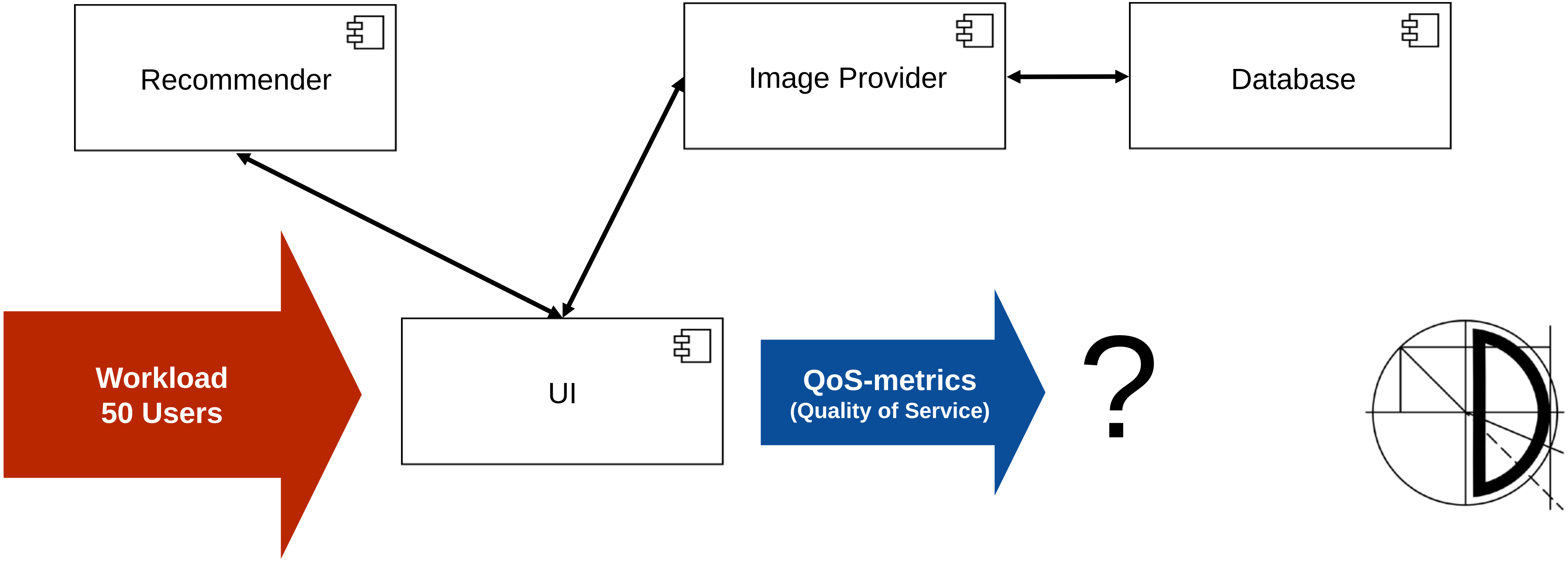
# Parametric Dependencies





# Parametric Dependencies

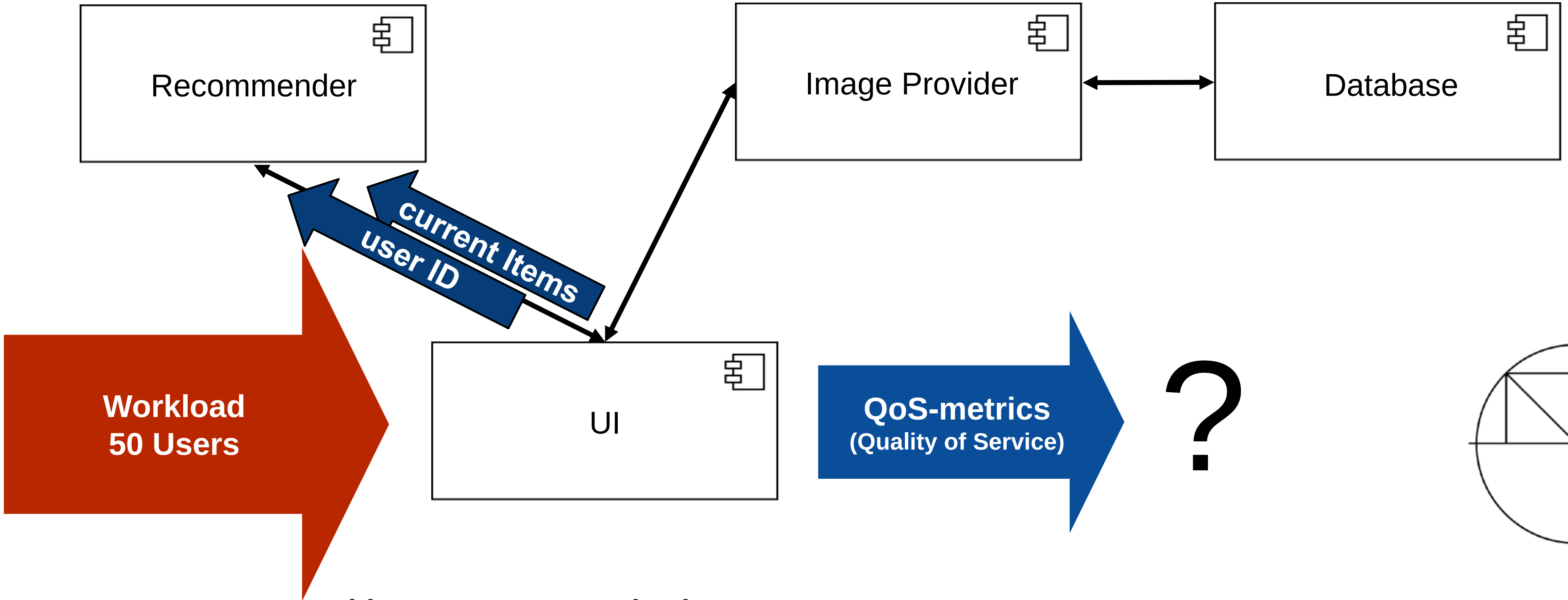




# Parametric Dependencies



- One parameter of performance models are parametric dependencies

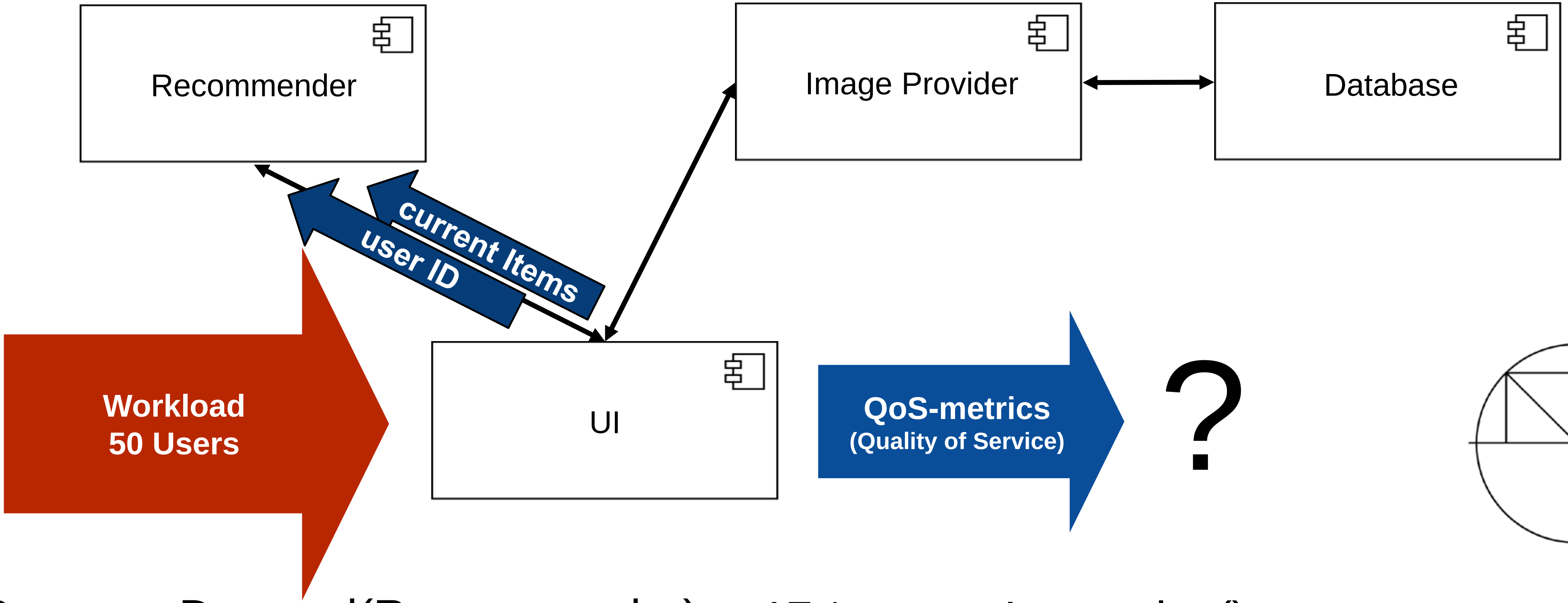



ResourceDemand(Recommender)
[milliseconds]

# Parametric Dependencies



- One parameter of performance models are parametric dependencies

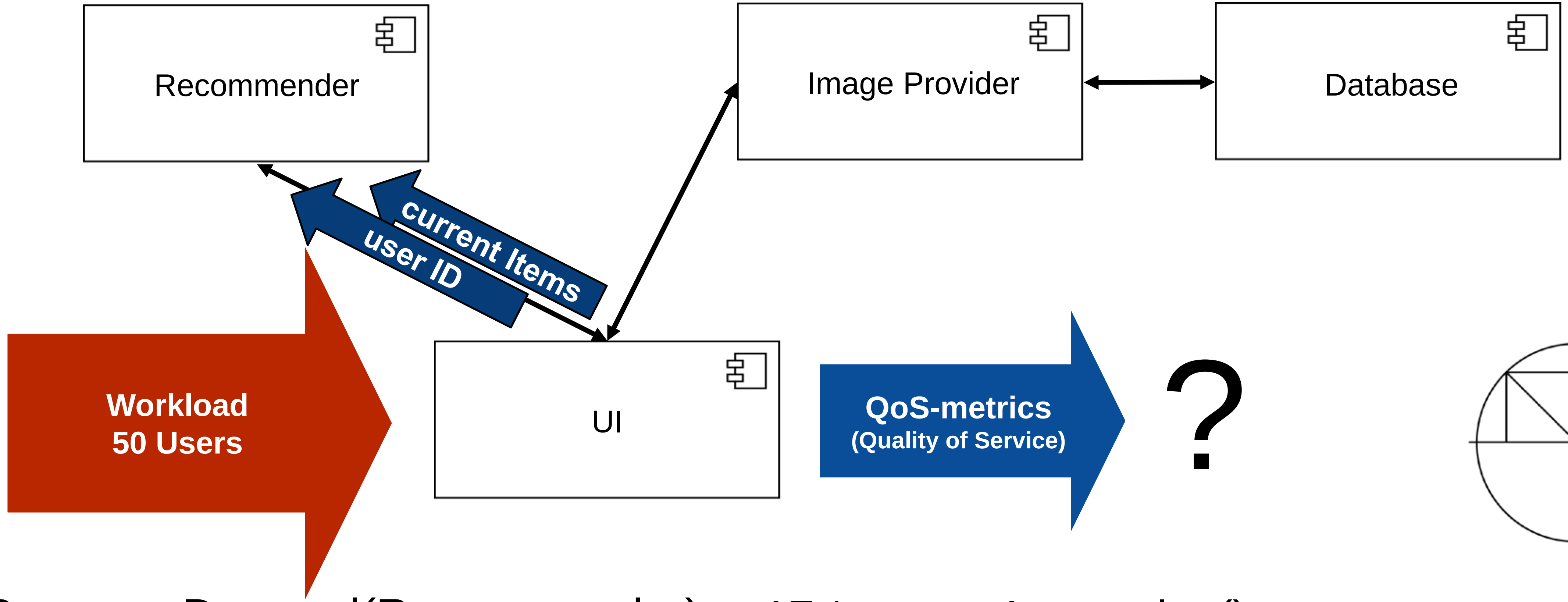



ResourceDemand(Recommender) = 17 * currentItems.size()
[milliseconds]

# Parametric Dependencies



- One parameter of performance models are parametric dependencies

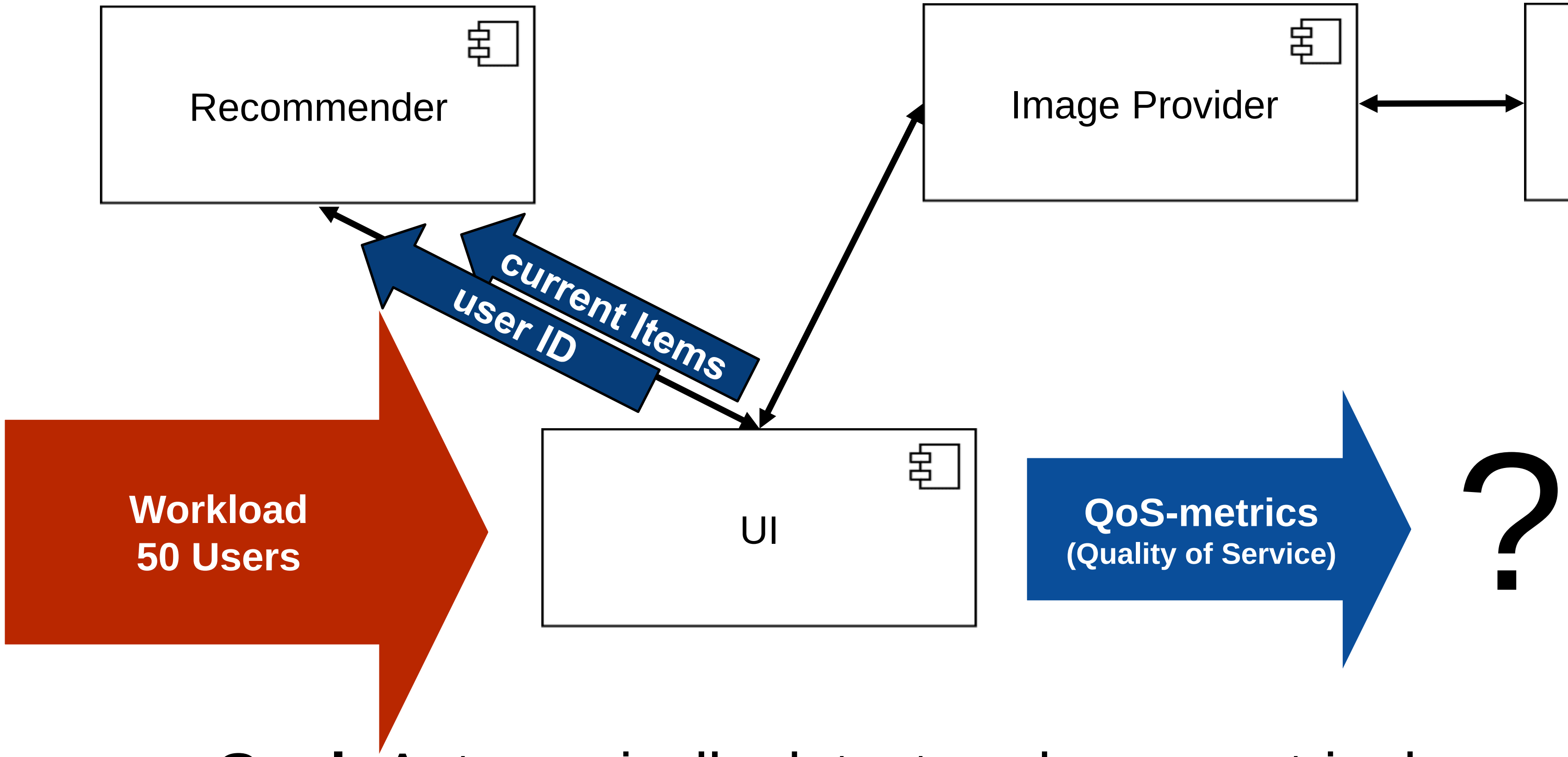



ResourceDemand(Recommender) = 17 * currentItems.size() + 0 * user ID
[milliseconds]

# Parametric Dependencies



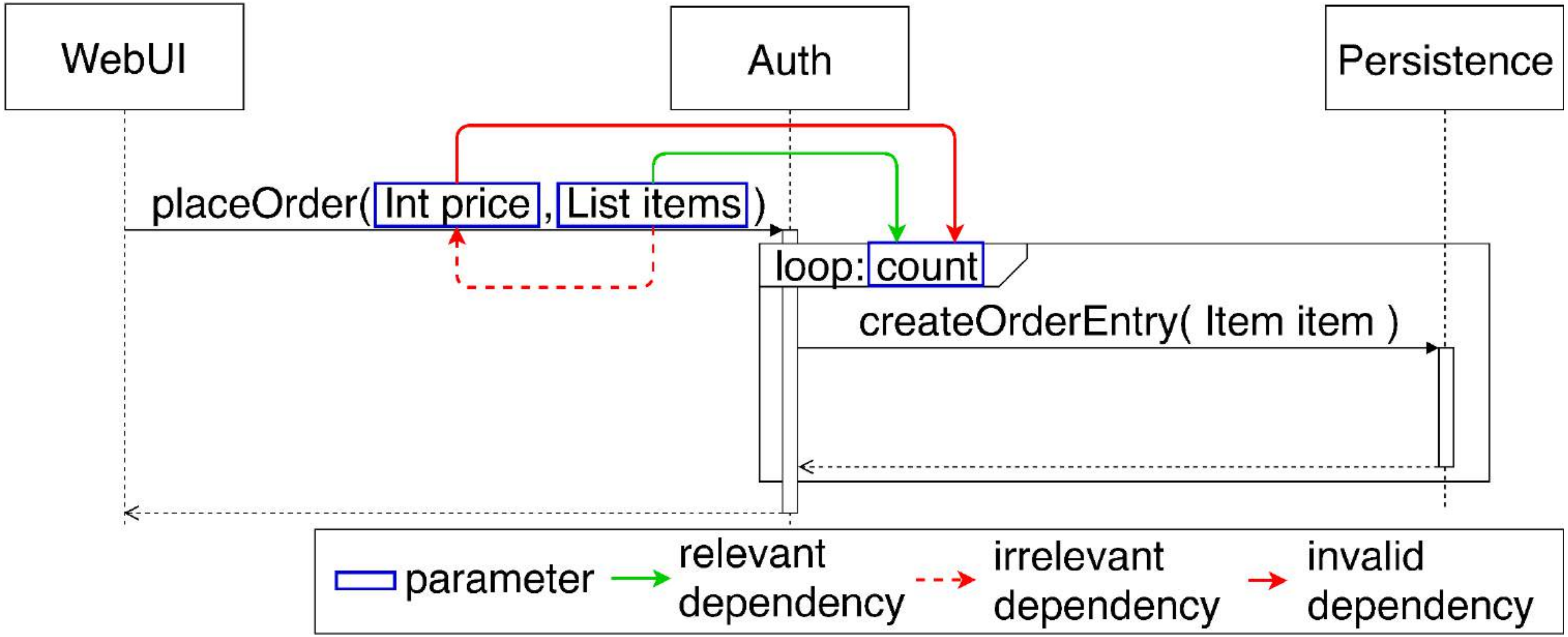
- One parameter of performance models are parametric dependencies




**Goal:** Autonomically detect such parametric dependencies

# Example




WebUI
Auth
Persistence
placeOrder( Int price , List items )
loop: count
createOrderEntry( Item item )
parameter
relevant dependency
irrelevant dependency
invalid dependency

# Related Work



- Krogmann et al. [KKR10] or Mazkatli and Koziolek [MK18] require source code for detection of dependencies. In contrast, our approch is solely based on monitoring data.

- Krogmann et al. [KKR10] or Mazkatli and Koziolek [MK18] require source code for detection of dependencies. In contrast, our approch is solely based on monitoring data.

Monitoring data

# Related Work



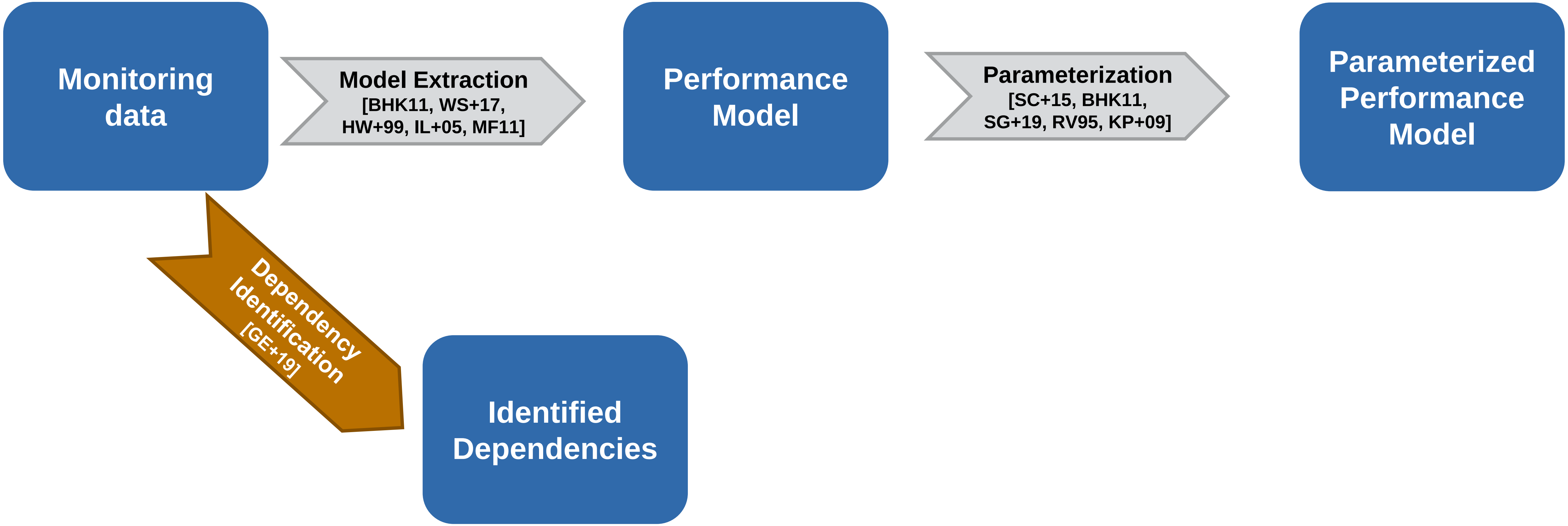
- Krogmann et al. [KKR10] or Mazkatli and Koziolek [MK18] require source code for detection of dependencies. In contrast, our approch is solely based on monitoring data.

**Monitoring data** → **Model Extraction** [BHK11, WS+17, HW+99, IL+05, MF11] → **Performance Model** → **Parameterization** [SC+15, BHK11, SG+19, RV95, KP+09] → **Parameterized Performance Model**

# Related Work



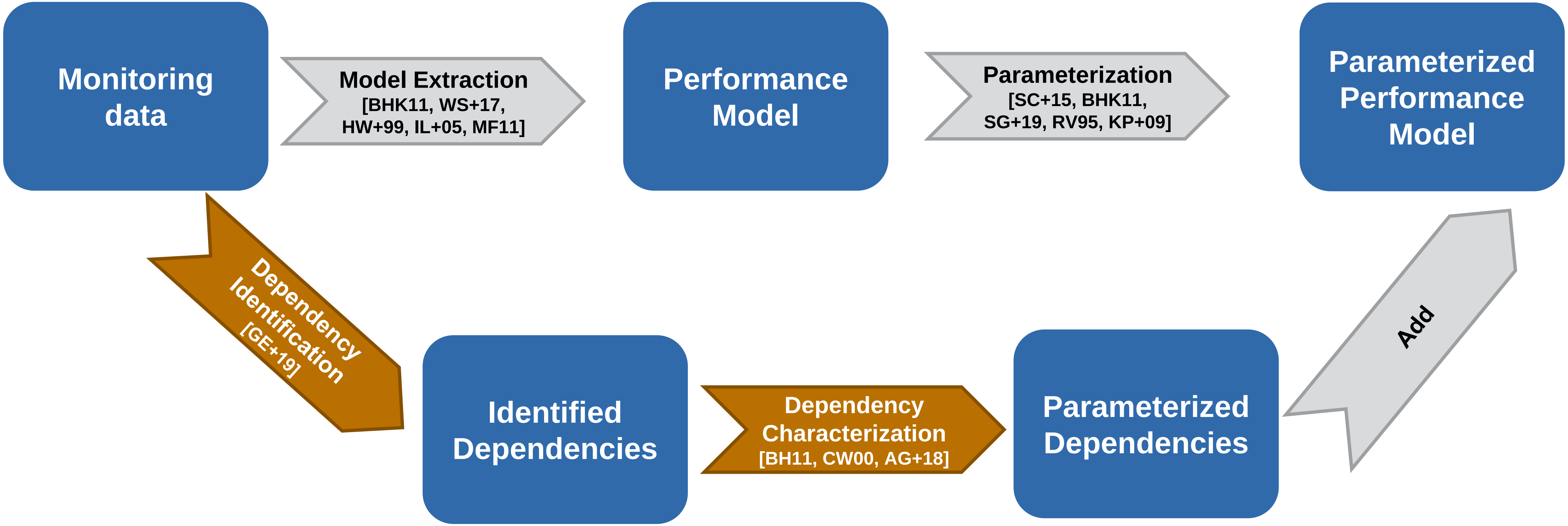
- Krogmann et al. [KKR10] or Mazkatli and Koziolek [MK18] require source code for detection of dependencies. In contrast, our approch is solely based on monitoring data.



# Related Work



- Krogmann et al. [KKR10] or Mazkatli and Koziolek [MK18] require source code for detection of dependencies. In contrast, our approch is solely based on monitoring data.



# In a nutshell

# In a nutshell



**Problem**

**Manual identification of parametric dependencies is not always possible, time-intensive and error-prone**

# In a nutshell



**Problem**

**Manual identification of parametric dependencies is not always possible, time-intensive and error-prone**

**Idea**

**Learning of dependencies using standard monitoring data collected during production**

# In a nutshell



**Problem**

**Manual identification of parametric dependencies is not always possible, time-intensive and error-prone**

**Idea**

**Learning of dependencies using standard monitoring data collected during production**

**Benefit**

**Increase model accuracy and expressiveness, additional step towards autonomic model learning**

# In a nutshell



| | |
|---|---|
| **Problem** | **Manual identification of parametric dependencies is not always possible, time-intensive and error-prone** |
| **Idea** | **Learning of dependencies using standard monitoring data collected during production** |
| **Benefit** | **Increase model accuracy and expressiveness, additional step towards autonomic model learning** |
| **Action** | **Use feature selection techniques for detecting, regression for characterizing the dependencies** |

# APPROACH

# Required monitoring information



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand

Identification parameters

# Required monitoring information



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand
  - Method signature
  - Entity

Identification parameters

Parameter-related information

# Required monitoring information



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand
  - Method signature
  - Entity
  - **Trace id**
  - Execution order index (**EOI**)
  - Execution stack size (**ESS**)

Identification parameters

Parameter-related information

Trace reconstruction

# Required monitoring information



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand
  - Method signature
  - Entity
  - **Trace id**
  - Execution order index (**EOI**)
  - Execution stack size (**ESS**)

Identification parameters

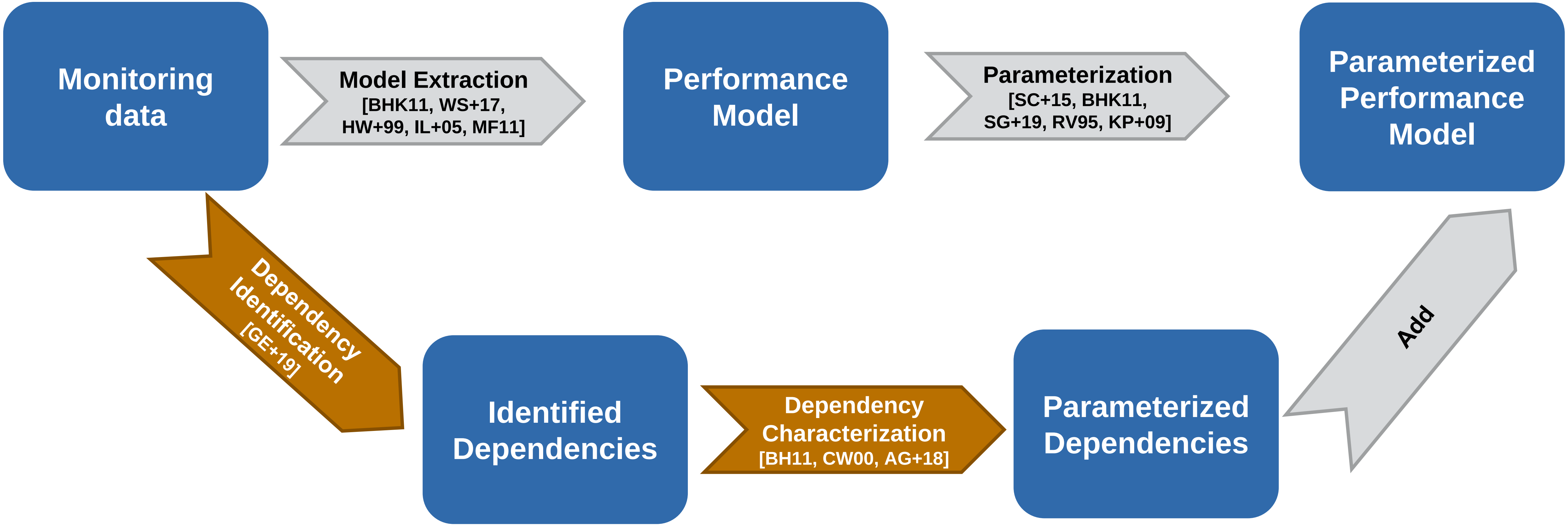
Parameter-related information

Trace reconstruction

- Enables reconstruction of call-path trace for resolving aggregations

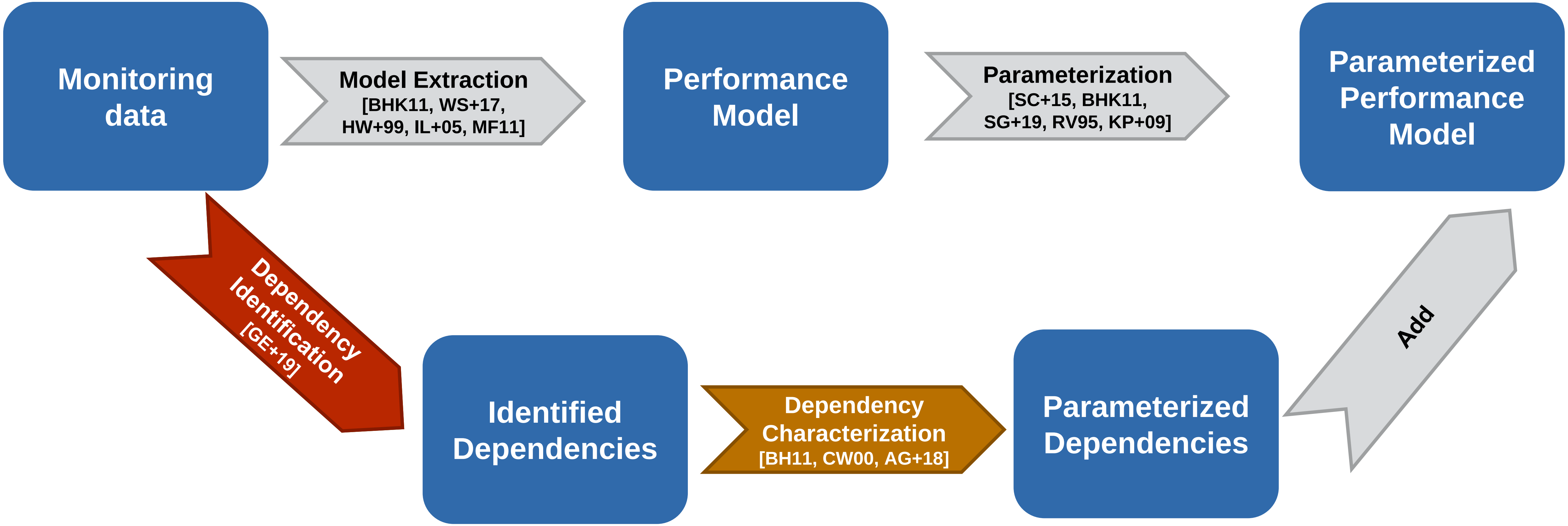
# Overview




Monitoring data
Model Extraction
[BHK11, WS+17, HW+99, IL+05, MF11]
Performance Model
Parameterization
[SC+15, BHK11, SG+19, RV95, KP+09]
Parameterized Performance Model
Dependency Identification
[GE+19]
Identified Dependencies
Dependency Characterization
[BH11, CW00, AG+18]
Parameterized Dependencies
Add

# Overview




Monitoring data
Model Extraction
[BHK11, WS+17, HW+99, IL+05, MF11]
Performance Model
Parameterization
[SC+15, BHK11, SG+19, RV95, KP+09]
Parameterized Performance Model
Dependency Identification
[GE+19]
Identified Dependencies
Dependency Characterization
[BH11, CW00, AG+18]
Parameterized Dependencies
Add

# Identification approaches



Monitoring Values | Model var.

$$\begin{pmatrix}3\\7\\3\\\dots\end{pmatrix} \quad \dots \quad \begin{pmatrix}8\\23\\95\\\dots\end{pmatrix} \begin{pmatrix}65\\32\\41\\\dots\end{pmatrix} \quad \Rightarrow \quad \begin{pmatrix}65\\32\\41\\\dots\end{pmatrix}$$

# Identification approaches



Monitoring Values → Features

Model var. → Target

$$\begin{pmatrix} 3 \\ 7 \\ 3 \\ \dots \end{pmatrix} \dots \begin{pmatrix} 8 \\ 23 \\ 95 \\ \dots \end{pmatrix} \begin{pmatrix} 65 \\ 32 \\ 41 \\ \dots \end{pmatrix} \Rightarrow \begin{pmatrix} 65 \\ 32 \\ 41 \\ \dots \end{pmatrix}$$

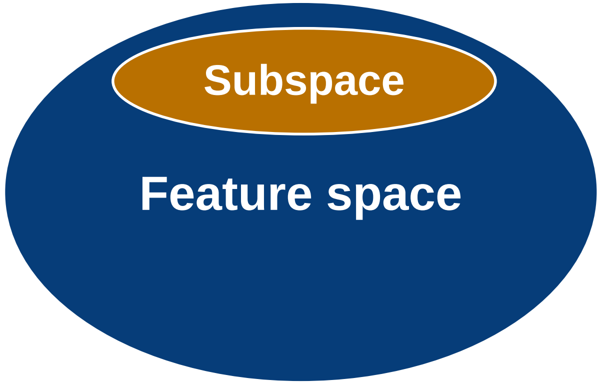
# Identification approaches



Monitoring Values → Features

Model var. → Target

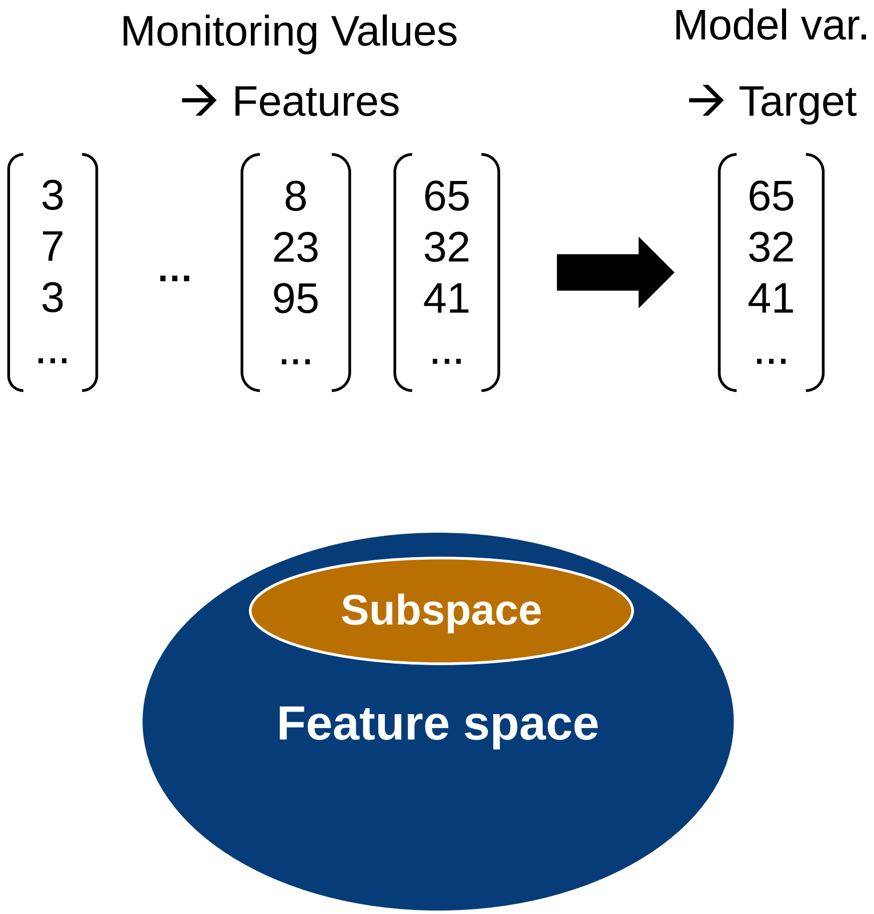
$$\begin{pmatrix} 3 \\ 7 \\ 3 \\ \dots \end{pmatrix} \dots \begin{pmatrix} 8 \\ 23 \\ 95 \\ \dots \end{pmatrix} \begin{pmatrix} 65 \\ 32 \\ 41 \\ \dots \end{pmatrix} \Rightarrow \begin{pmatrix} 65 \\ 32 \\ 41 \\ \dots \end{pmatrix}$$



# Identification approaches



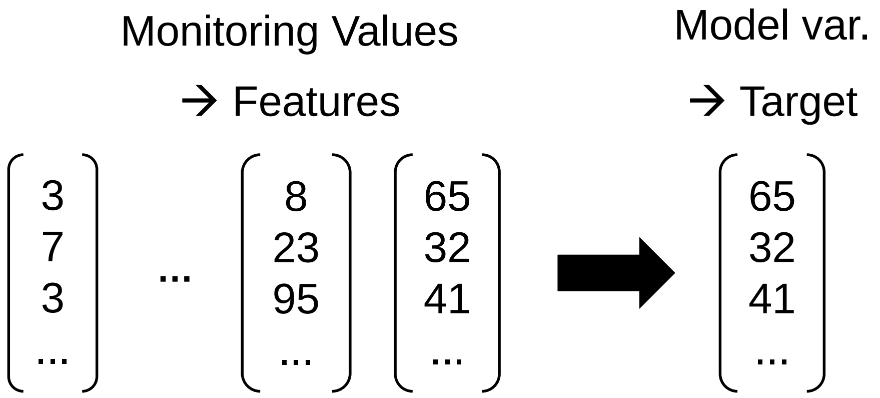
- Embedded
  - Evaluate feature importance during training
  - Selection based on comparison with „noise feature“
  - Algorithm: Random forest [H95]



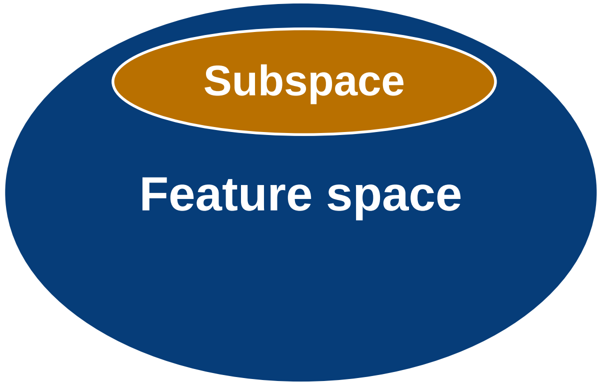
# Identification approaches



- Embedded
  - Evaluate feature importance during training
  - Selection based on comparison with „noise feature“
  - Algorithm: Random forest [H95]

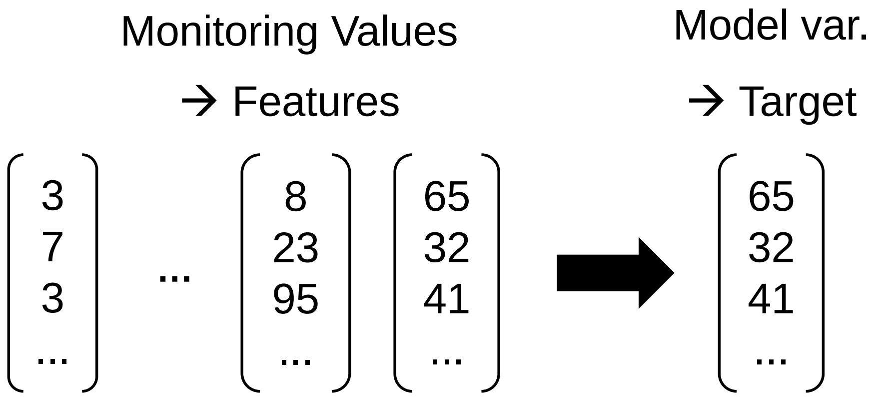



- Wrapper
  - Selection based on accuracy error for a feature subset, compared with a baseline regressor
  - Algorithm: M5 trees [Q+92] and Linear regression

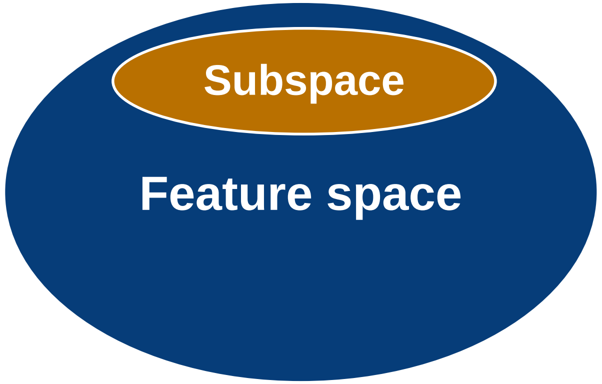


# Identification approaches





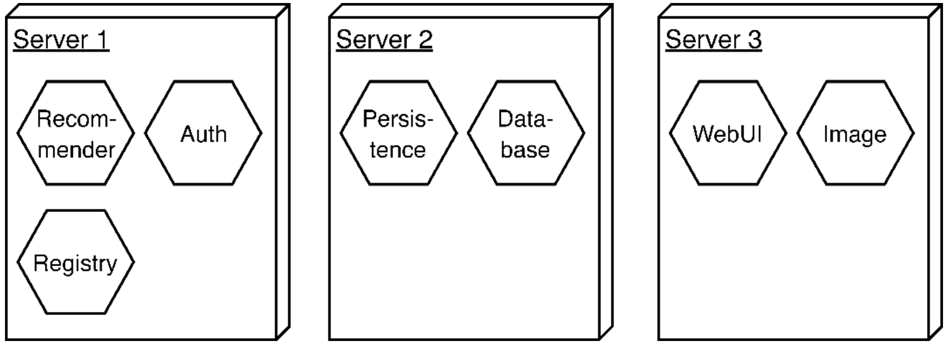

- Embedded
  - Evaluate feature importance during training
  - Selection based on comparison with „noise feature“
  - Algorithm: Random forest [H95]
- Wrapper
  - Selection based on accuracy error for a feature subset, compared with a baseline regressor
  - Algorithm: M5 trees [Q+92] and Linear regression
- Filter: Correlation-based
  - Pearson product-moment correlation coefficient (PPMCC)
  - Selection based on threshold for correlation

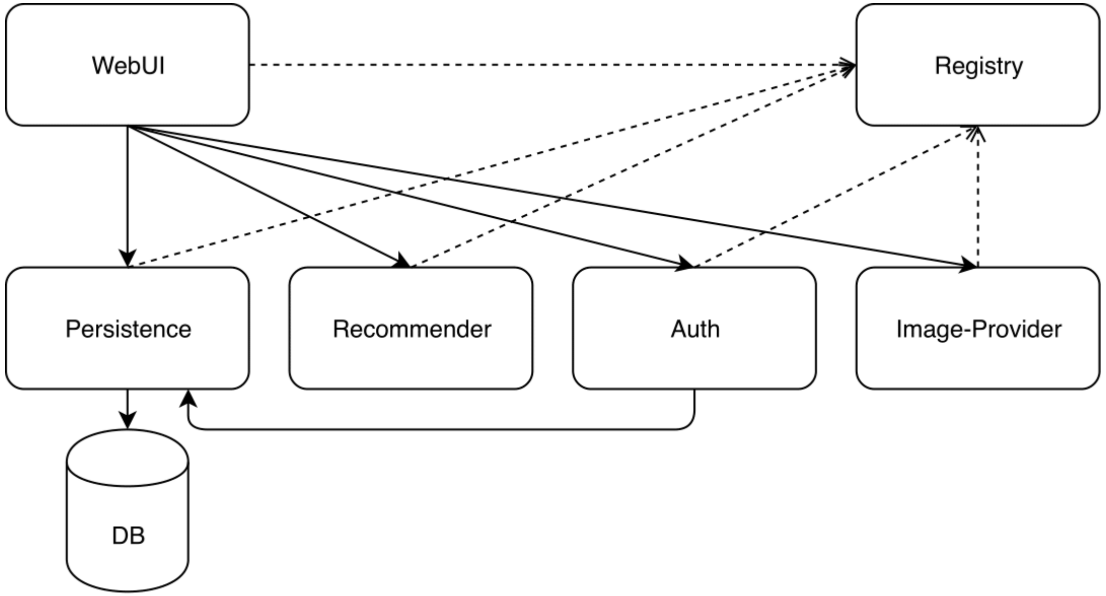


# Evaluation



- Distributed deployment of TeaStore [vKE+18] application

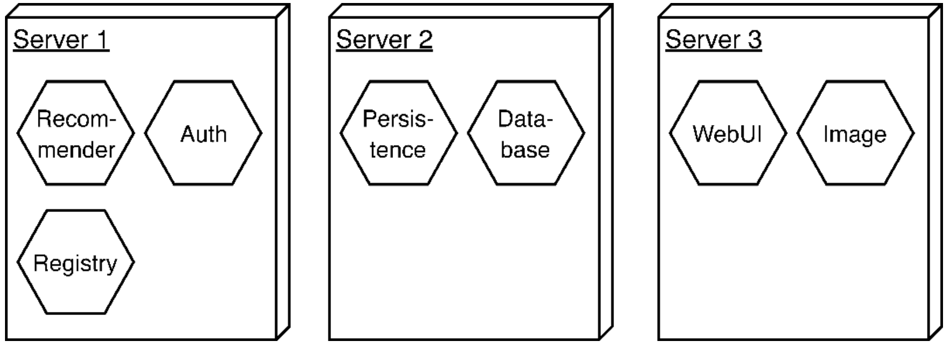


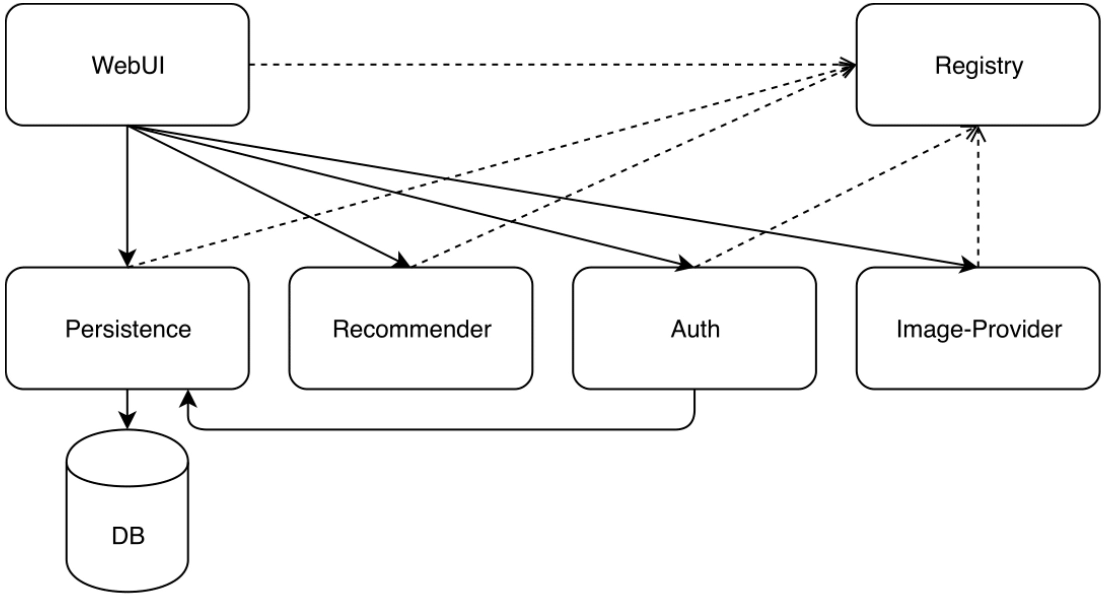



# Evaluation



- Distributed deployment of TeaStore [vKE+18] application

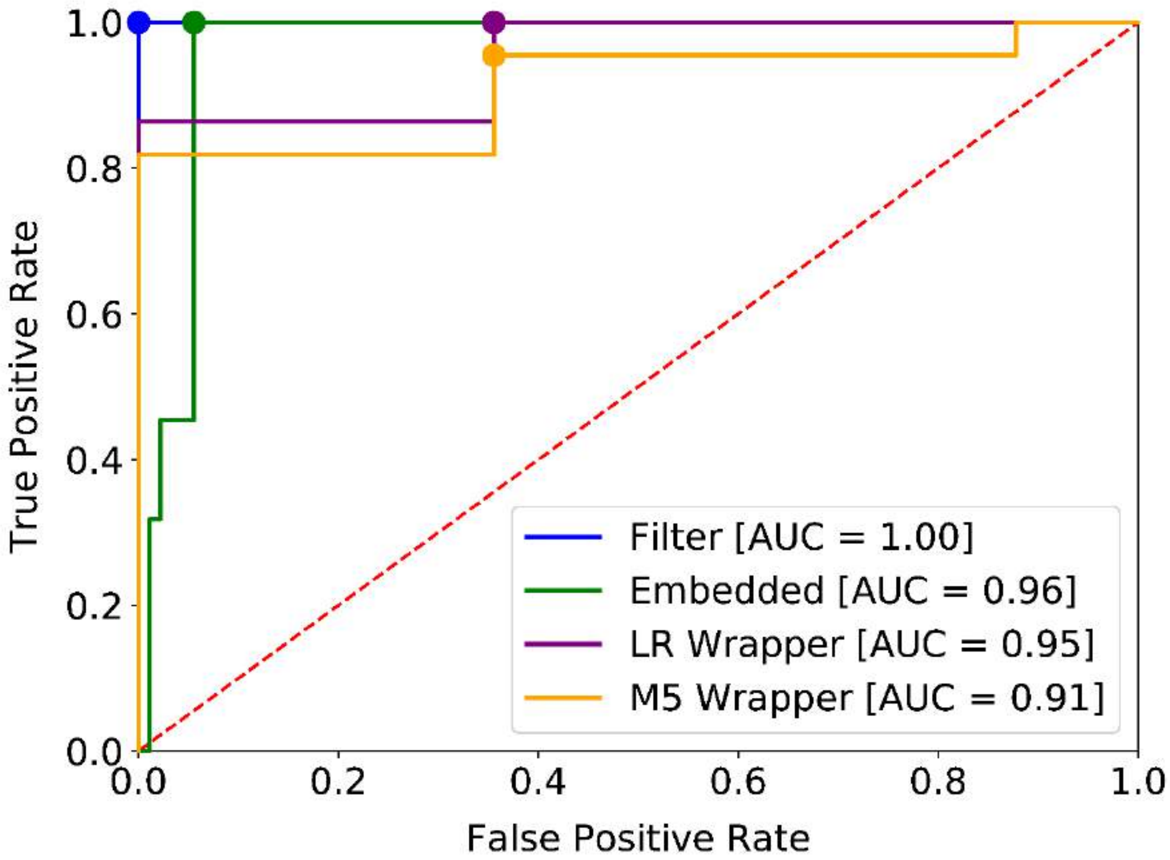



- Locust as load driver with typical behavior of customer
  - Login & logout
  - Browse for products
  - Add products to cart
  - Checkout cart

# Selection Thresholds






- Filter approach outperforms other approaches
- Results are threshold-independent

# Filter Application



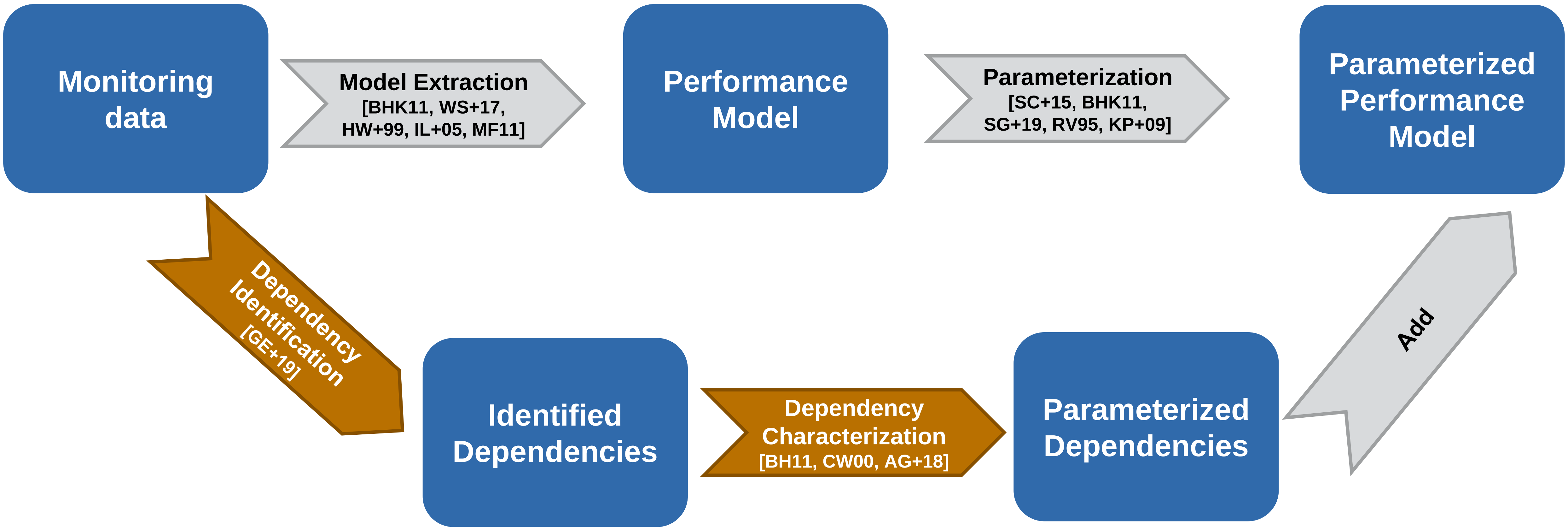
| Filtering Step | Relevant | Irrelevant | Invalid | Total |
|---|---|---|---|---|
| None | 11 | 94 | 5 | 110 |
| Identical (1) | 11 | 45 | 5 | 61 |
| (1) + Correlating (2) | 11 | 35 | 1 | 47 |
| (1) + (2) + Graph-based (3) | 11 | 8 | 1 | 20 |

**In total, 86 irrelvant and and 4 invalid dependencies are deleted.**

**This results in a precision (11 relevant to 1 invalid) of 91.7 %.**

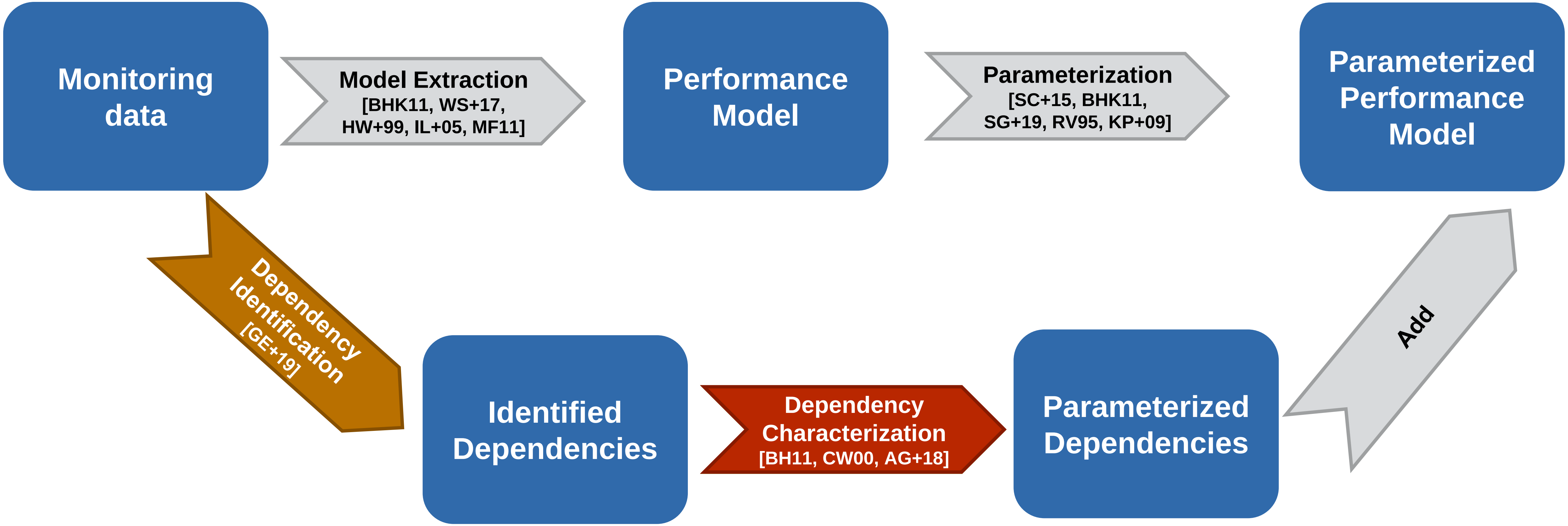
# Overview




Monitoring data
Model Extraction
[BHK11, WS+17, HW+99, IL+05, MF11]
Performance Model
Parameterization
[SC+15, BHK11, SG+19, RV95, KP+09]
Parameterized Performance Model
Dependency Identification
[GE+19]
Identified Dependencies
Dependency Characterization
[BH11, CW00, AG+18]
Parameterized Dependencies
Add

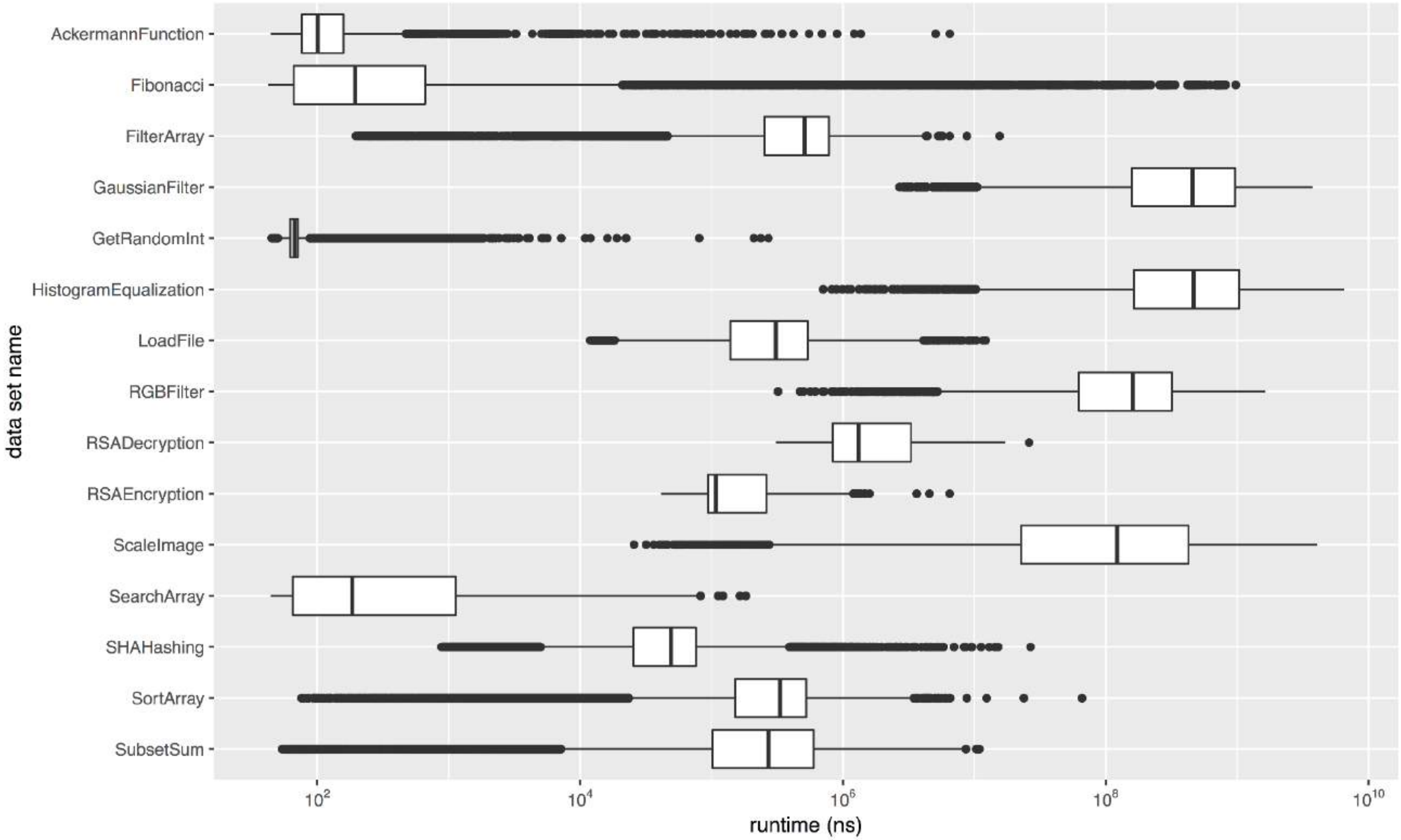
# Overview




Monitoring data
Model Extraction
[BHK11, WS+17, HW+99, IL+05, MF11]
Performance Model
Parameterization
[SC+15, BHK11, SG+19, RV95, KP+09]
Parameterized Performance Model
Dependency Identification
[GE+19]
Identified Dependencies
Dependency Characterization
[BH11, CW00, AG+18]
Parameterized Dependencies
Add

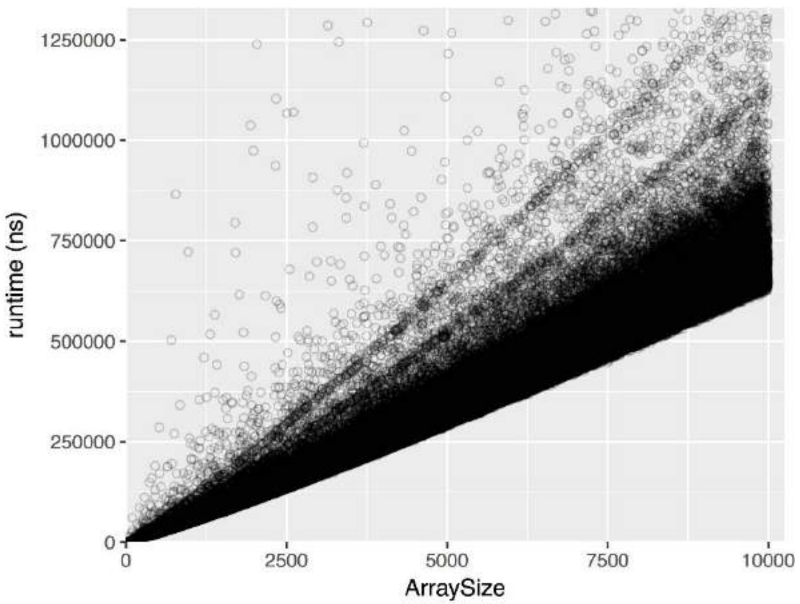
# Dataset Characteristics I



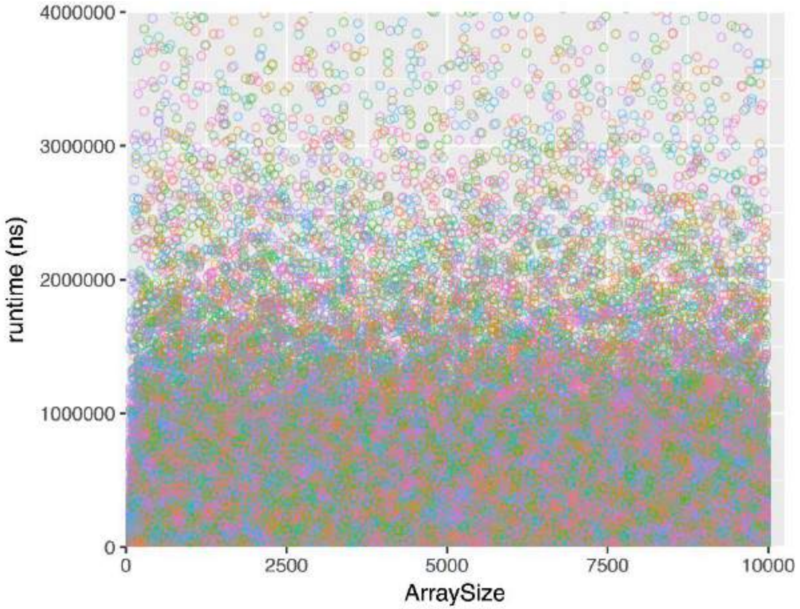


# Dataset Characteristics II



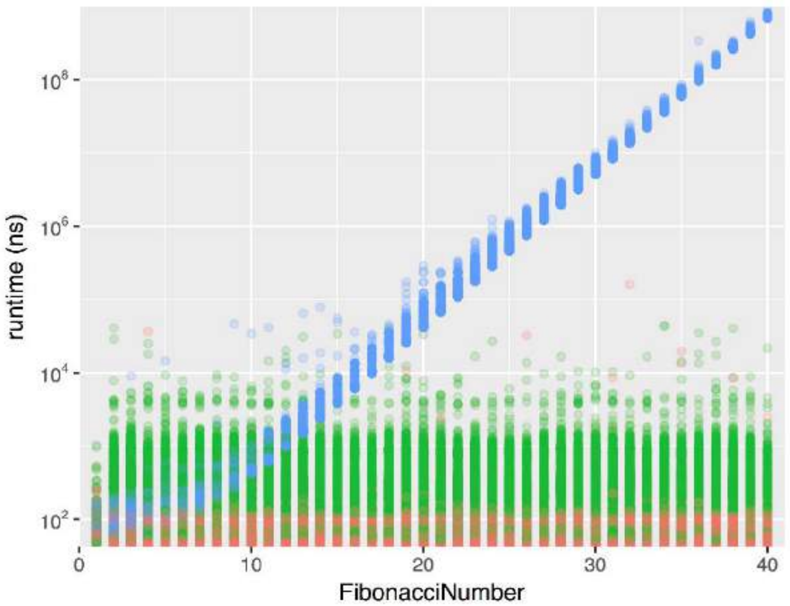



SortArray




SubsetSum

Colors encode defined sum

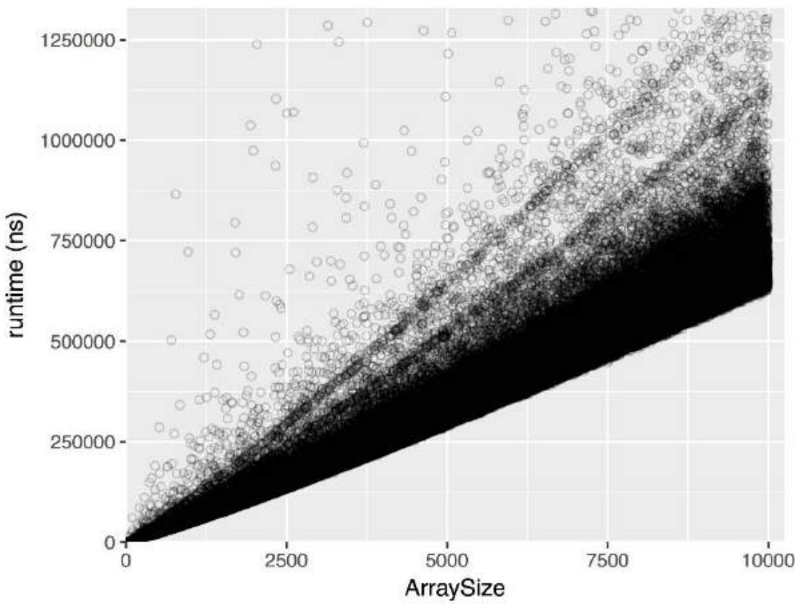



Fibonacci

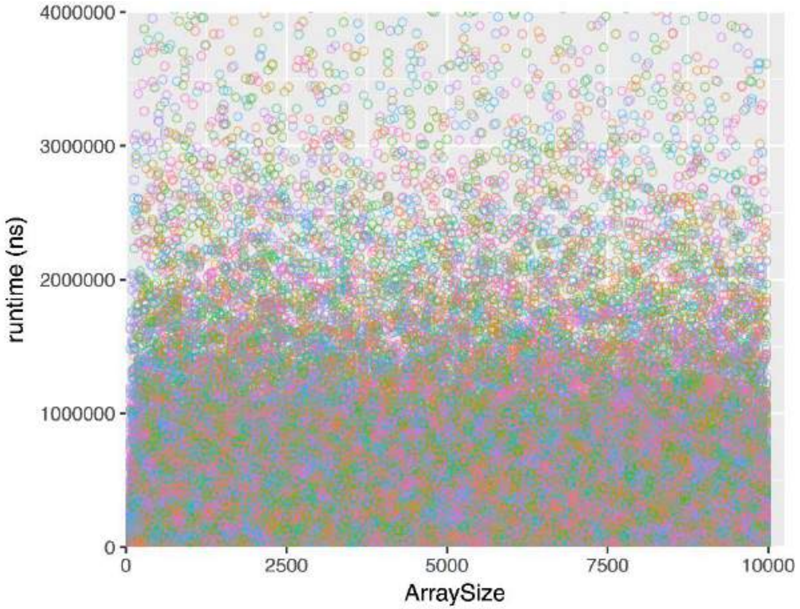
Recursive
Optimized recursive
Iterative

# Dataset Characteristics II



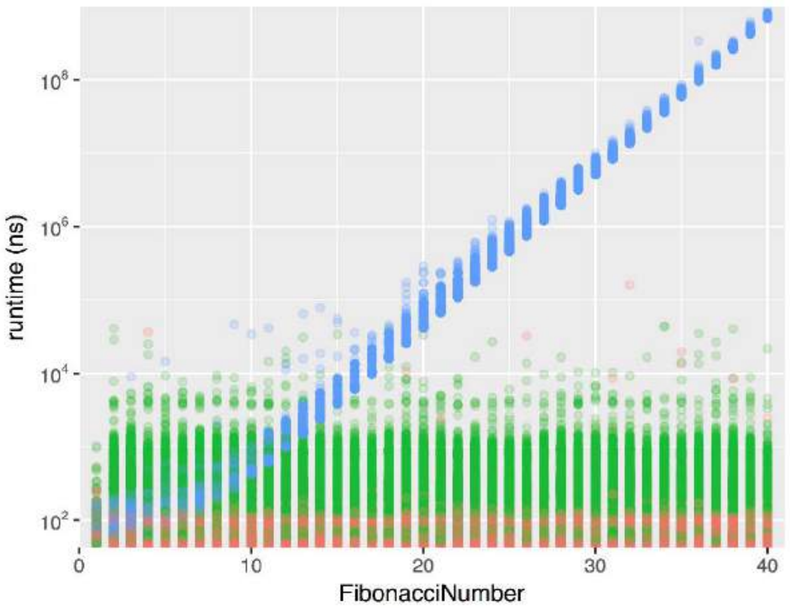



SortArray




SubsetSum

Colors encode defined sum




Fibonacci

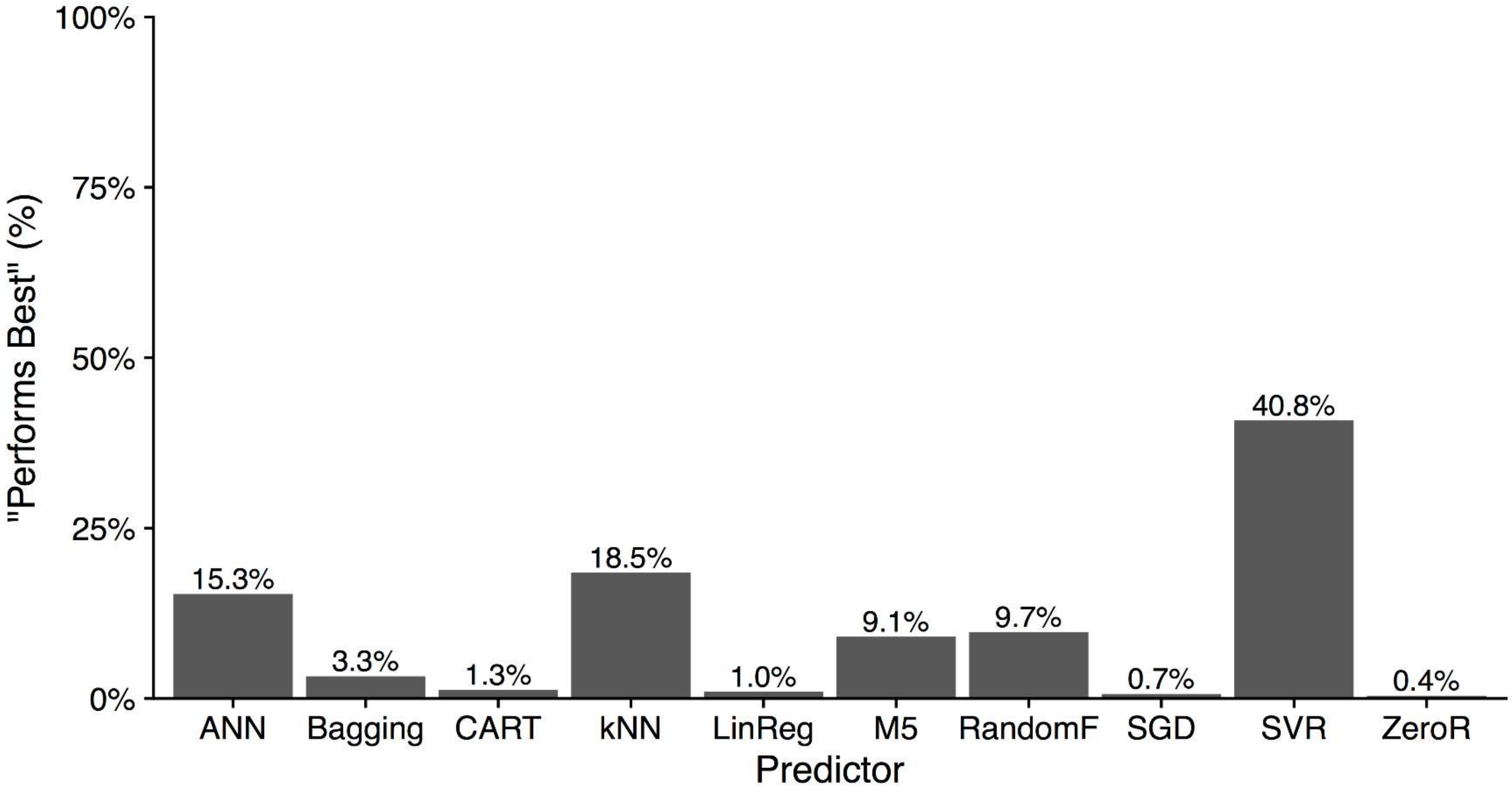
Recursive
Optimized recursive
Iterative

**The datasets are diverse and varying in terms of number and types of parameters, distribution of runtime (resource demand) and type of dependency.**

# No Free Lunch

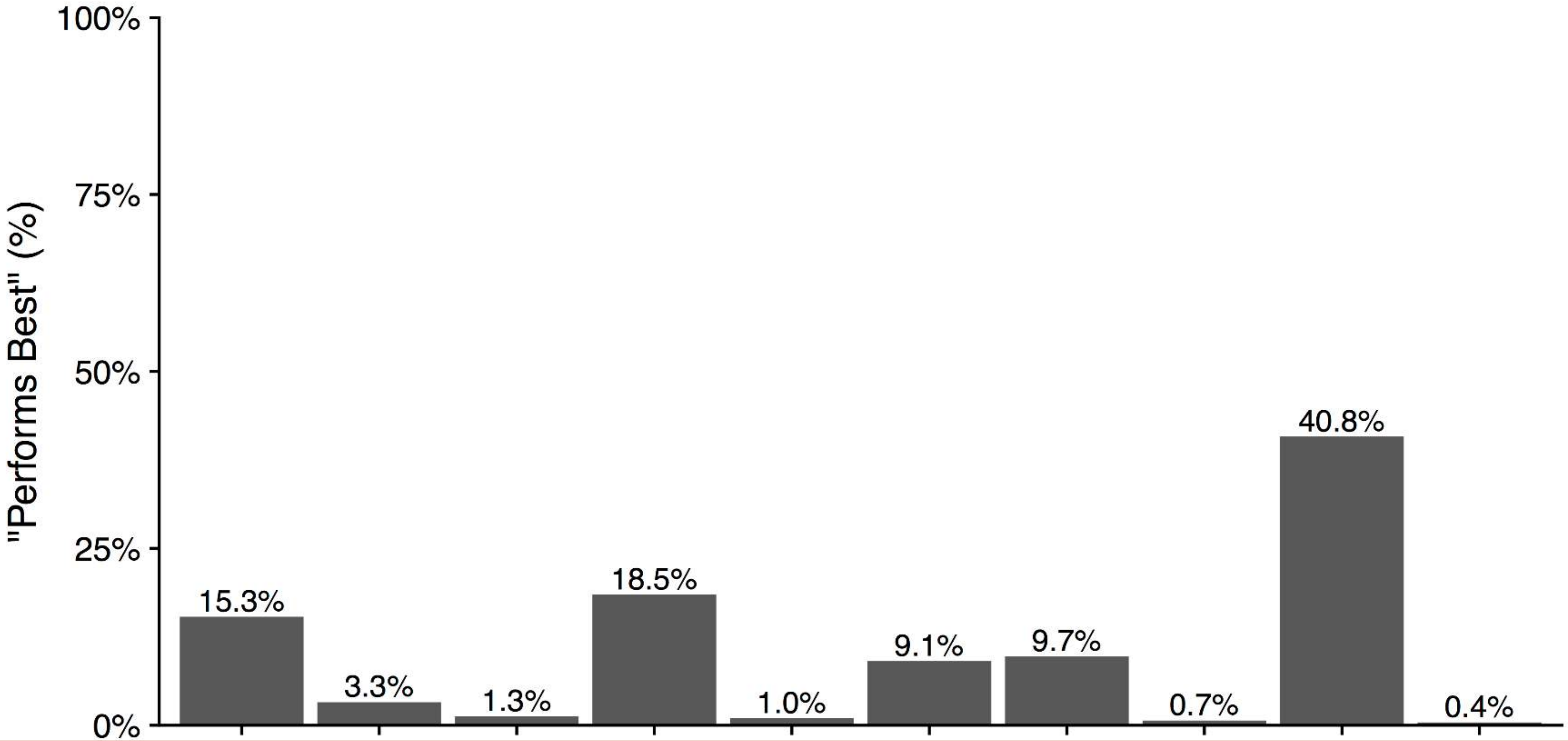
# No Free Lunch





# No Free Lunch



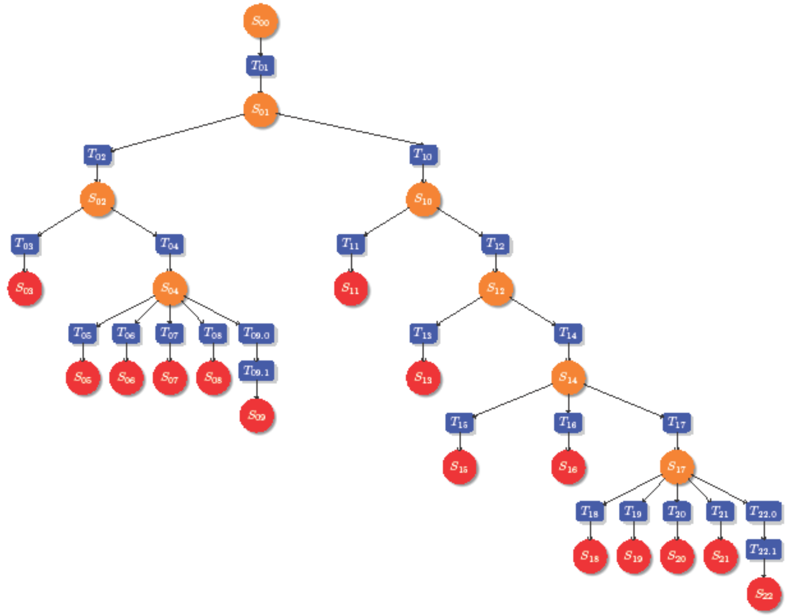



**No Free Lunch for ML approaches! [8]**
**We need a meta-classifier to select the appropriate algorithm.**

# Meta-Classifier



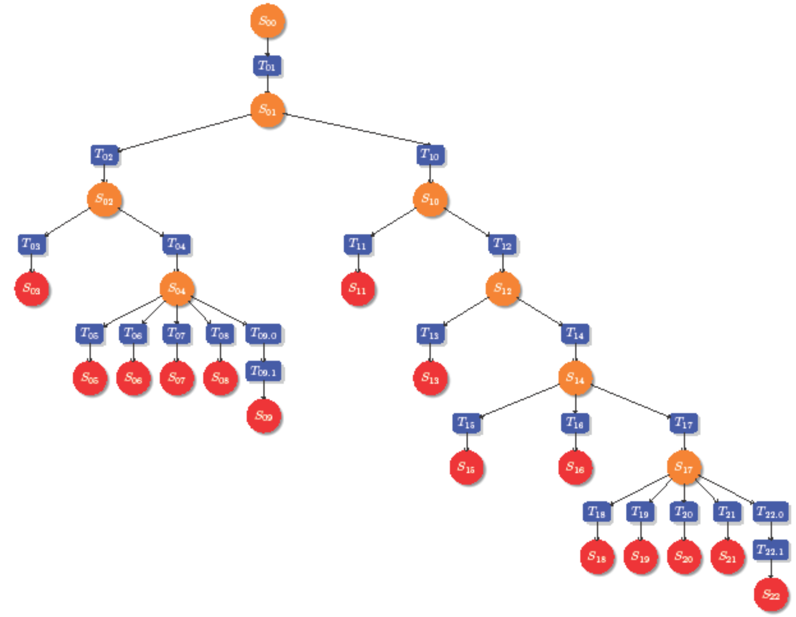
- Using Classification and Regression Trees (CART) to train a Decision Tree on the following features:



# Meta-Classifier



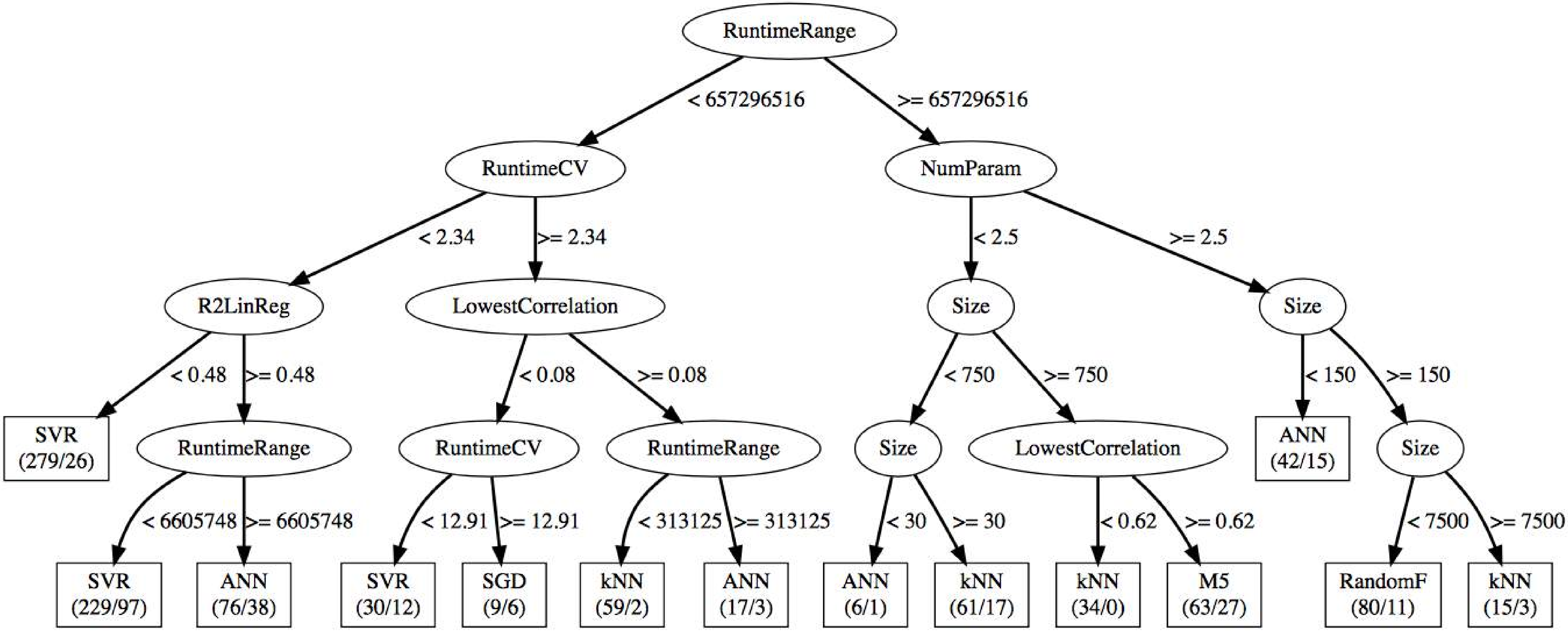
- Using Classification and Regression Trees (CART) to train a Decision Tree on the following features:



  - Number of training instances (Size)
  - Number of parameters (NumParam)
  - Range of runtime values (RuntimeRange)
  - Coefficient of variance of runtime (RuntimeCV)
  - Highest linear correlation between any input parameter and runtime (HighestCorrelation)
  - Lowest linear correlation between any input parameter and runtime (LowestCorrelation)
  - Coefficient of determination (R2) (R2LinReg)

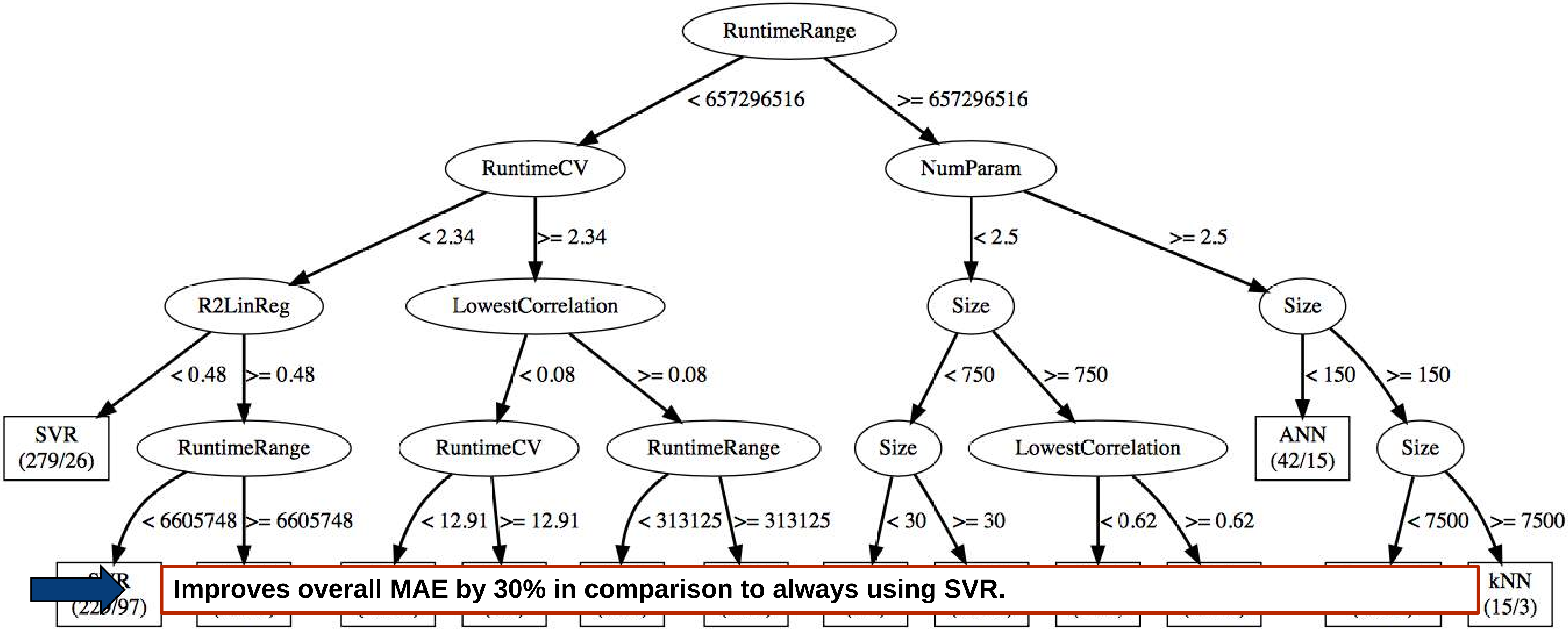
# Meta-Classifier II





# Meta-Classifier II






**Improves overall MAE by 30% in comparison to always using SVR.**

# OPEN CHALLENGES

# Monitoring Challenges



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand

  Identification parameters

  - Method signature
  - Entity

  Parameter-related information

  - **Trace id**
  - Execution order index (**EOI**)
  - Execution stack size (**ESS**)

  Trace reconstruction

- Enables reconstruction of call-path trace for resolving aggregations

# Monitoring Challenges



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand
  - Method signature
  - Entity
  - **Trace id**
  - Execution order index (**EOI**)
  - Execution stack size (**ESS**)

**What are the important features of each parameter?**

**How can the features be extracted?**

Parameter-related information

Trace reconstruction

- Enables reconstruction of call-path trace for resolving aggregations

# Monitoring Challenges



- Monitoring data per invocation through Kieker [vHWH12] monitoring
  - Parameter values and types
  - Return value and type
  - Resource demand
  - Method signature
  - Entity
  - **Trace id**
  - Execution order index (**EOI**)
  - Execution stack size (**ESS**)

**What are the important features of each parameter?**

**How can the features be extracted?**

Parameter-related information

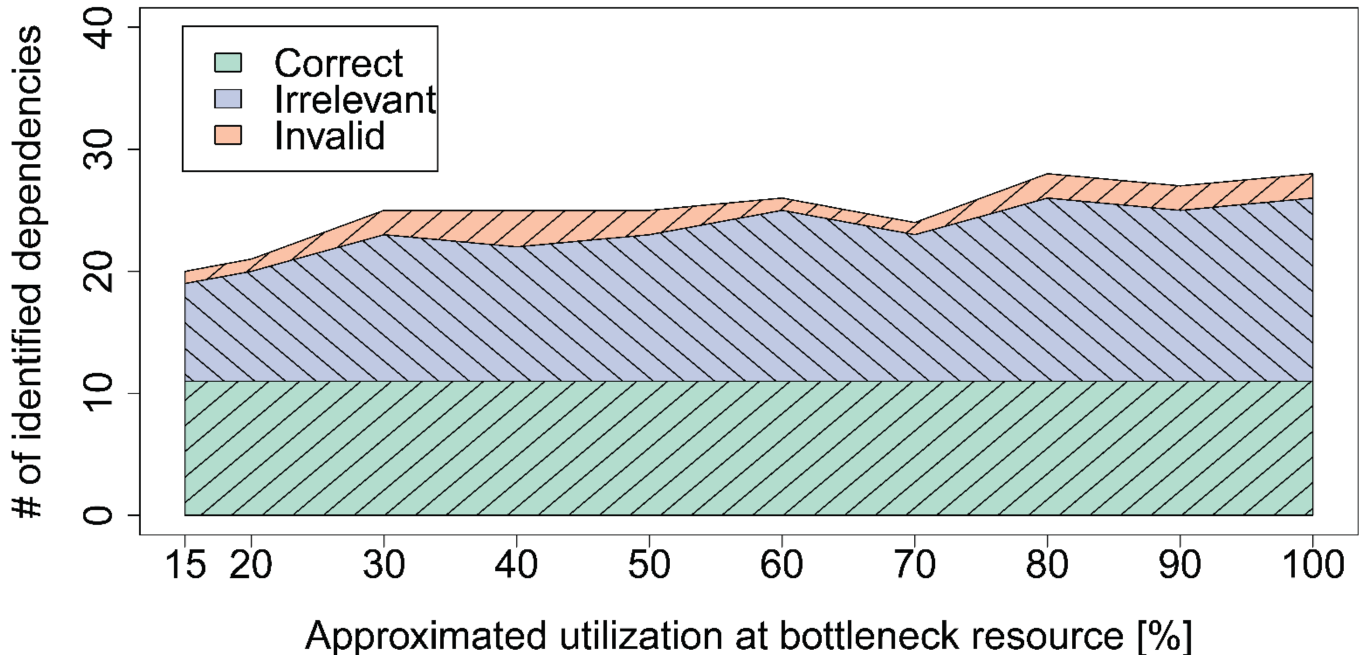
**We can only observe the response time.**

**How can the resource demands be measured?**

- Enables reconstruction of call-path trace for resolving aggregations

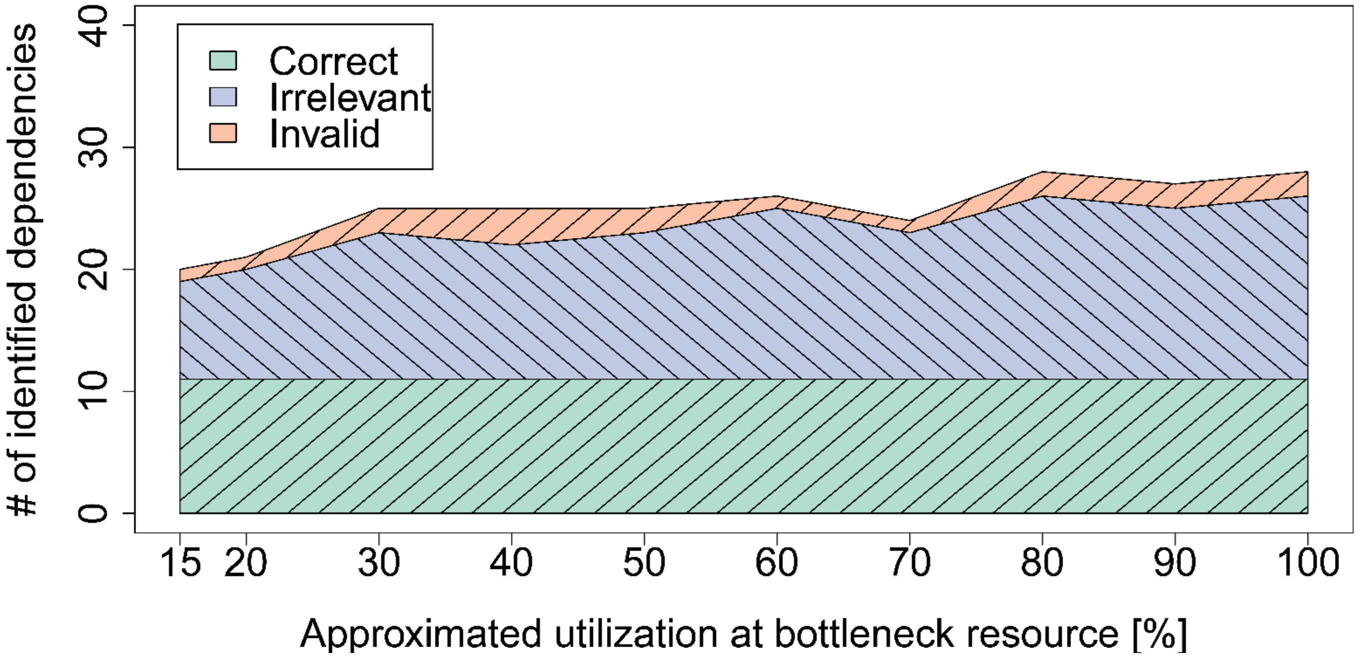
# Stability in higher load scenarios





# Stability in higher load scenarios

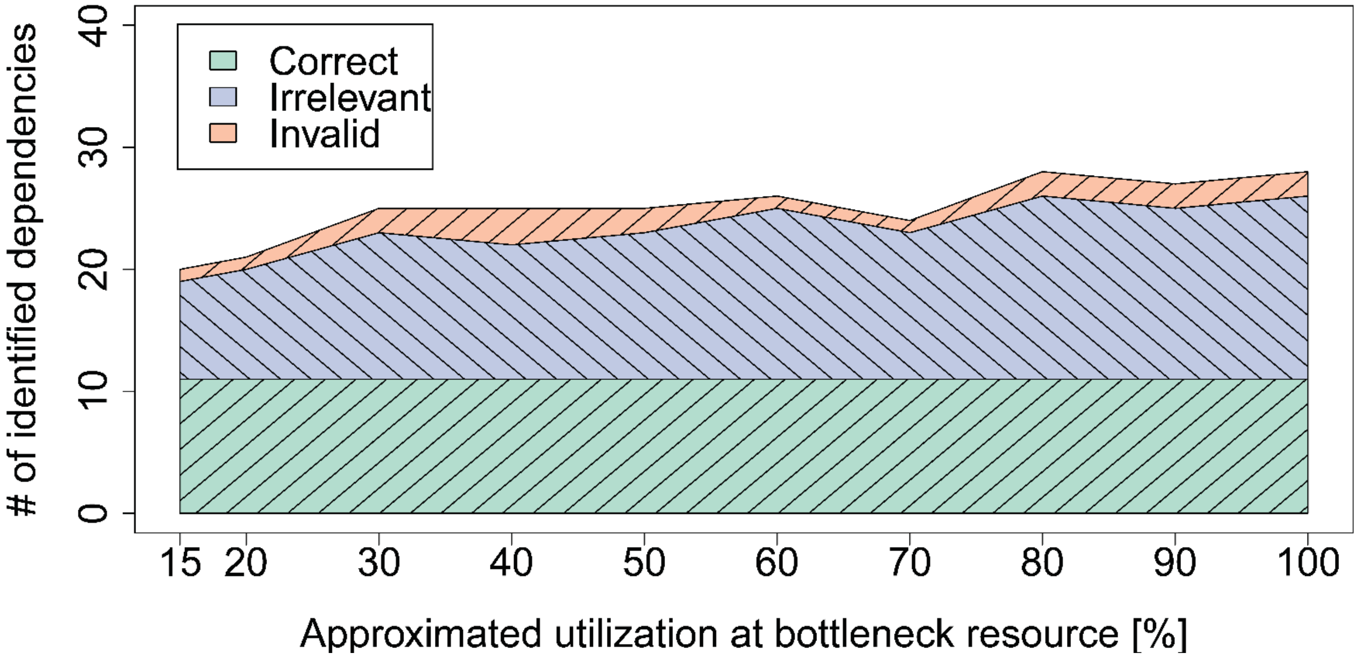





- Higher loads lead to less accuracy; however the effect is light

# Stability in higher load scenarios

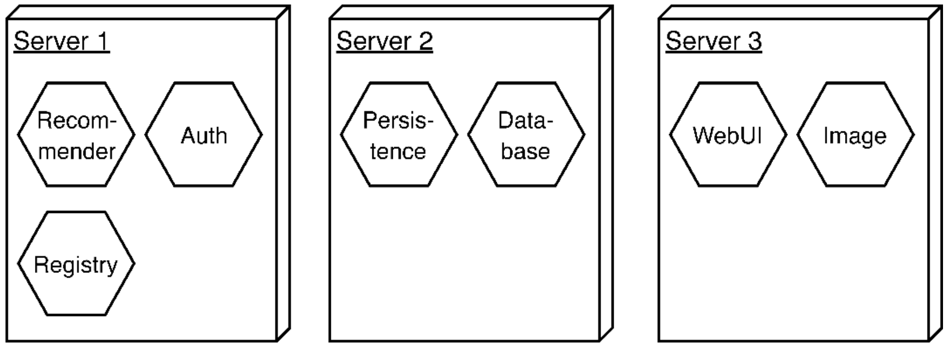


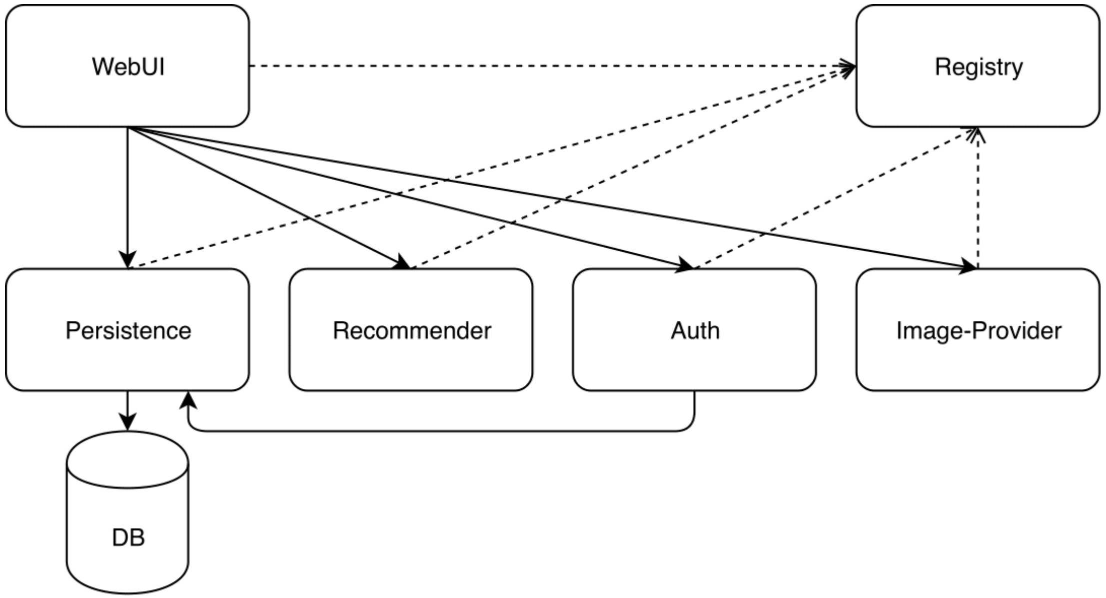



- Higher loads lead to less accuracy; however the effect is light
- All relevant dependencies are still found

# Evaluation Challenges

- Distributed deployment of TeaStore [vKE+18] application



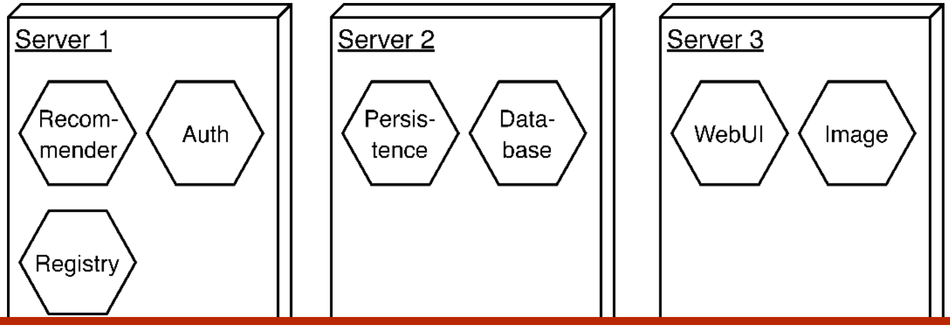




- Locust as load driver with typical behavior of customer
  - Login & logout
  - Browse for products
  - Add products to cart
  - Checkout cart

# Evaluation Challenges



- Distributed deployment of TeaStore [vKE+18] application

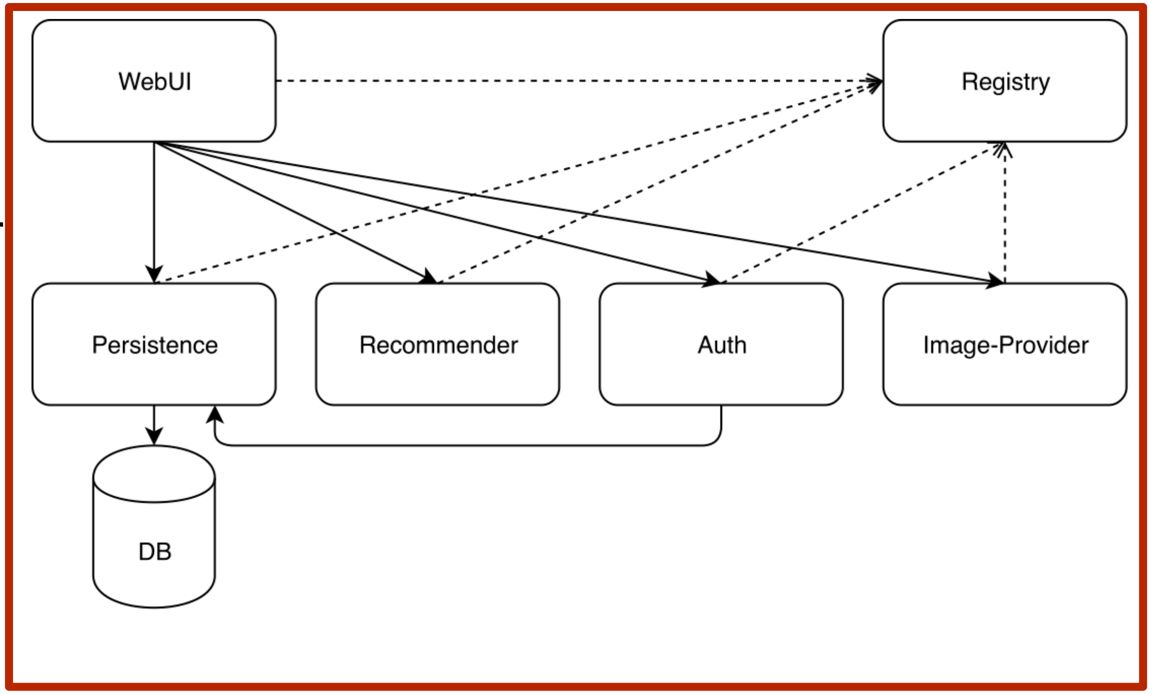



**We need dependencies as gold standard.**

**How can they be achieved?**

**Comparison with other paradigms required?**

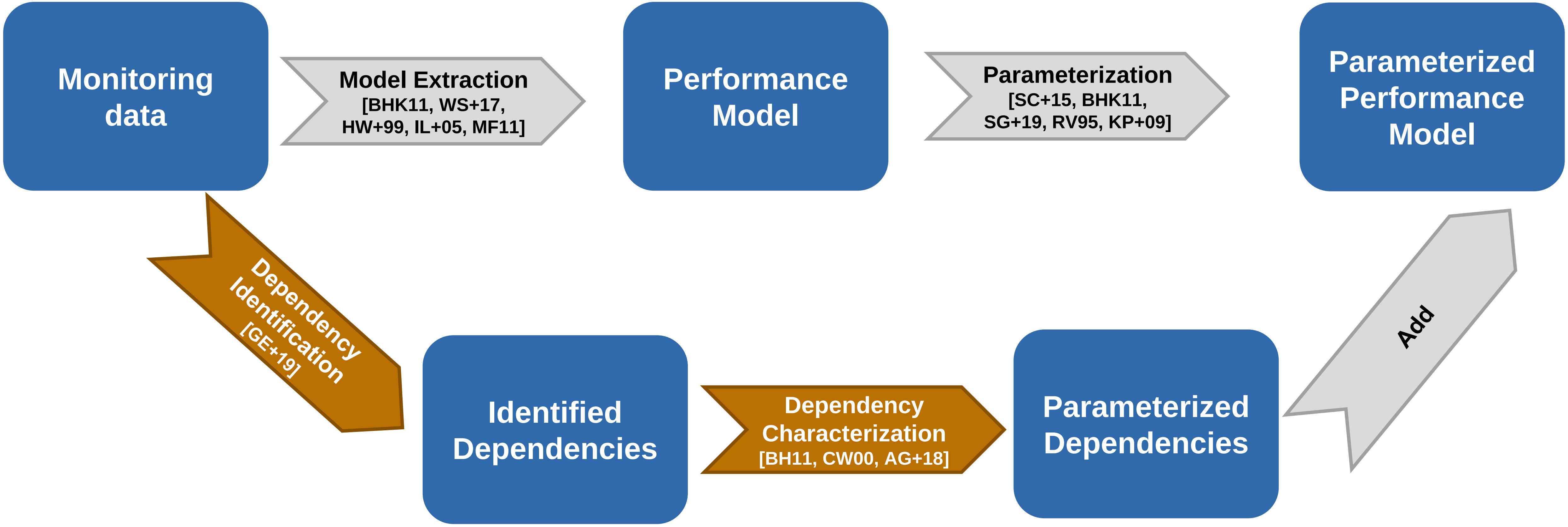
er




  - Add products to cart
  - Checkout cart

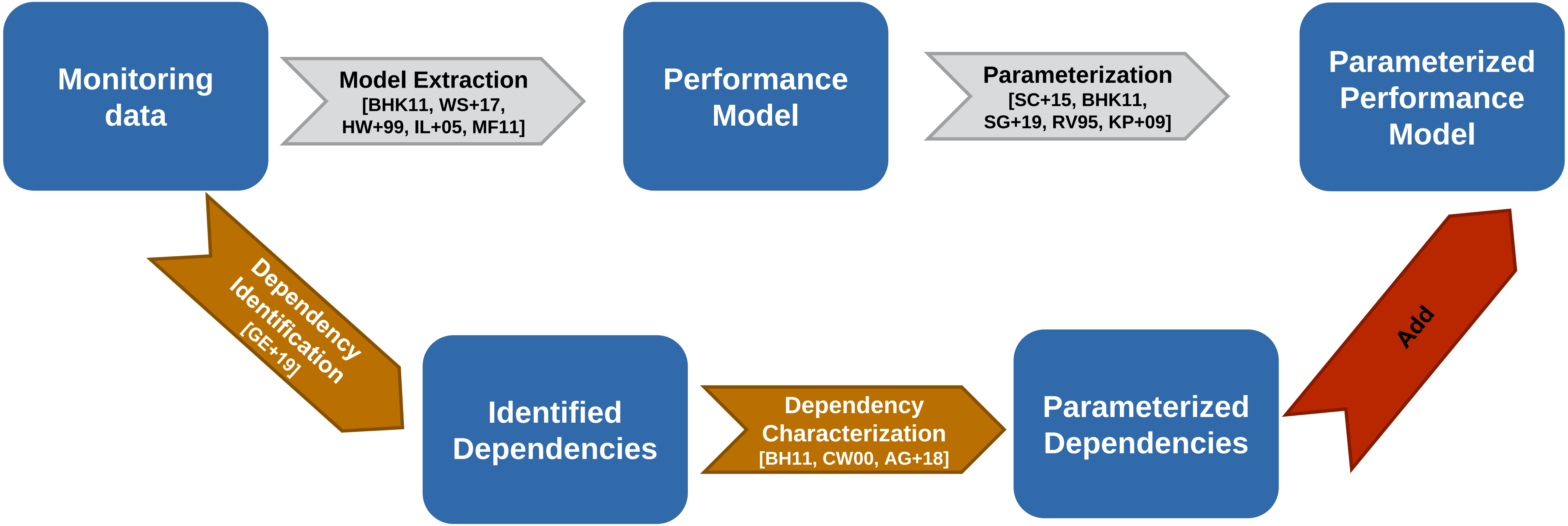
# Integration Challenges





# Integration Challenges





# Conclusion

# Conclusion



**Problem**

**Manual identification of parametric dependencies is not always possible, time-intensive and error-prone**

# Conclusion



| | |
|---|---|
| **Problem** | **Manual identification of parametric dependencies is not always possible, time-intensive and error-prone** |
| **Idea** | **Learning of dependencies using standard monitoring data collected during production** |

# Conclusion



**Problem**: Manual identification of parametric dependencies is not always possible, time-intensive and error-prone

**Idea**: Learning of dependencies using standard monitoring data collected during production

**Benefit**: Increase model accuracy and expressiveness, additional step towards autonomic model learning

# Conclusion



| | |
|---|---|
| **Problem** | **Manual identification of parametric dependencies is not always possible, time-intensive and error-prone** |
| **Idea** | **Learning of dependencies using standard monitoring data collected during production** |
| **Benefit** | **Increase model accuracy and expressiveness, additional step towards autonomic model learning** |
| **Action** | **Use feature selection techniques for detecting, regression for characterizing the dependencies** |

# References


- [KKR10] K. Krogmann, M. Kuperberg, and R. Reussner. “Using genetic search for reverse engineering of parametric behavior models for performance prediction”. In: IEEE Transactions on Software Engineering 36.6 (2010), pp. 865–877.
- [MK18] M. Mazkatli and A. Koziolek. “Continuous Integration of Performance Model”. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. ICPE ’18. Berlin, Germany: ACM, 2018, pp. 153–158.
- [BHK11] F. Brosig, N. Huber, and S. Kounev, “Automated extraction of architecture-level performance models of distributed component-based systems,” in 26th IEEE/ACM International Conference On Automated Software Engineering (ASE 2011), Oread, Lawrence, Kansas, November 2011.
- [WS+17] J. Walter, C. Stier, H. Koziolek, and S. Kounev, “An Expandable Extraction Framework for Architectural Performance Models,” in Proceedings of the 3rd International Workshop on Quality-Aware DevOps (QUDOS’17). ACM, April 2017.
- [HW+99] C. E. Hrischuk, C. M. Woodside, J. A. Rolia, and R. Iversen, “Tracebased load characterization for generating performance software models,” IEEE Trans. Softw. Eng., vol. 25, no. 1, pp. 122–135, Jan. 1999.
- [IL+05] T. A. Israr, D. H. Lau, G. Franks, and M. Woodside, “Automatic generation of layered queuing software performance models from commonly available traces,” in Proceedings of the 5th International Workshop on Software and Performance, ser. WOSP ’05. New York, USA: ACM, 2005, pp. 147–158.
- [MF11] A. Mizan and G. Franks, “An automatic trace based performance evaluation model building for parallel distributed systems,” SIGSOFT Softw. Eng. Notes, vol. 36, no. 5, pp. 61–72, Sep. 2011.
- [CW00] M. Courtois and M. Woodside, “Using regression splines for software performance analysis,” in Proceedings of the 2nd International Workshop on Software and Performance, 2000, pp. 105–114.
- [AG+18] V. Ackermann, J. Grohmann, S. Eismann, and S. Kounev, “Black-box learning of parametric dependencies for performance models,” in 13th International Workshop on Models@run.time (MRT), co-located with ACM/IEEE 21st International Conference on Model Driven Engineering Languages and Systems (MODELS 2018), ser. CEUR Workshop Proceedings, October 2018.

# References


- [SG+19] S. Spinner, J. Grohmann, S. Eismann, and S. Kounev, “Online model learning for self-aware computing infrastructures,” Journal of Systems and Software, vol. 147, pp. 1 – 16, 2019.
- [SC+15] S. Spinner, G. Casale, F. Brosig, and S. Kounev, “Evaluating Approaches to Resource Demand Estimation,” Perform. Evaluation, vol. 92, pp. 51 – 71, October 2015.
- [RV95] J. Rolia and V. Vetland, “Parameter estimation for performance models of distributed application systems,” in CASCON ’95. IBM Press, 1995, p. 54.
- [KP+09] S. Kraft, S. Pacheco-Sanchez, G. Casale, and S. Dawson, “Estimating service resource consumption from response time measurements,” in VALUETOOLS ’09, 2009, pp. 1–10.
- [vHWH12] A. van Hoorn, J. Waller, and W. Hasselbring, “Kieker: A framework for application performance monitoring and dynamic software analysis,” in Proceedings of the 3rd joint ACM/SPEC International Conference on Performance Engineering, 2012, pp. 247–248.
- [WF+16] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques, 4th ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016.
- [H95] T. K. Ho, “Random decision forests,” in Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1), ser. ICDAR ’95. Washington, DC, USA: IEEE Computer Society, 1995.
- [Q+92] J. R. Quinlan et al., “Learning with continuous classes,” in 5th Australian joint conference on artificial intelligence, vol. 92. Singapore, 1992, pp. 343–348.
- [vKE+18] J. von Kistowski, S. Eismann, N. Schmitt, A. Bauer, J. Grohmann, and S. Kounev, “Teastore: A micro-service reference application for benchmarking, modeling and resource management research,” in Proceedings of the 26th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, ser. MASCOTS ’18, September 2018.
- [GE+19] Johannes Grohmann, Simon Eismann, Sven Elflein, Manar Mazkatli, Jóakim von Kistowski, and Samuel Kounev. Detecting Parametric Dependencies for Performance Models Using Feature Selection Techniques. In Proceedings of the 27th IEEE International Symposium on the Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, Rennes, France, October 2019, MASCOTS '19.

# Thank you for your attention! ☺

https://se.informatik.uni-wuerzburg.de/